TI-PROGRAMMABLE 57

INTRODUCTION TO PROGRAMMING

TEXAS INSTRUMENTS

KEY INDEX

This indexed keyboard provides a quick page reference to the description of each key.

2nd 1-3, 2-1	INV 2-1	log 2-14 ln$x$ 2-14	C.t 2-3 CE 2-3	CLR 2-3
D.MS 2-19 LRN 1-2, 3-2	P→R 2-20 $x \rightleftarrows t$ 2-22, 3-8	sin 2-17 x^2 2-12	cos 2-17 $\sqrt{x}$ 2-12	tan 2-17 $1/x$ 2-12
Pause 1-3, 3-3 SST 3-14	Ins 3-15 STO 1-7, 2-7	Exc 2-8 RCL 2-7, 3-8	Prd 2-8 SUM 2-8	π 2-12 y^x 2-13
Nop 3-15 BST 3-15	Del 3-16 EE 2-10	FIX 2-11 (2-6	Int 2-15) 2-6, 4-11	$\lvert x \rvert$ 2-12 ÷ 2-4
Dsz 3-6 GTO 3-5, 3-15	7	8	9	Deg 2-16 × 2-4
$x=t$ 3-8 SBR 3-11	4	5	6	Rad 2-16 − 2-4
$x \geq t$ 3-10 RST 1-3, 3-2	1	2	3	Grad 2-16 + 2-4
Lbl 3-4 R/S 1-6, 3-3	0	Σ+ 2-22 • 2-4	$\bar{x}$ 2-23 +/− 2-4	$\sigma 2$ 2-23 = 2-4

IMPORTANT

Record the serial number from the bottom of the unit and purchase date in the space below. The serial number is identified by the words "SERIAL NO." on the bottom case. Always reference this information in any correspondence.

TI PROGRAMMABLE-57

Model No. ______ Serial No. ______ Purchase Date ______

©Copyright, Texas Instruments, 1977

INTRODUCTION
TO
PROGRAMMING

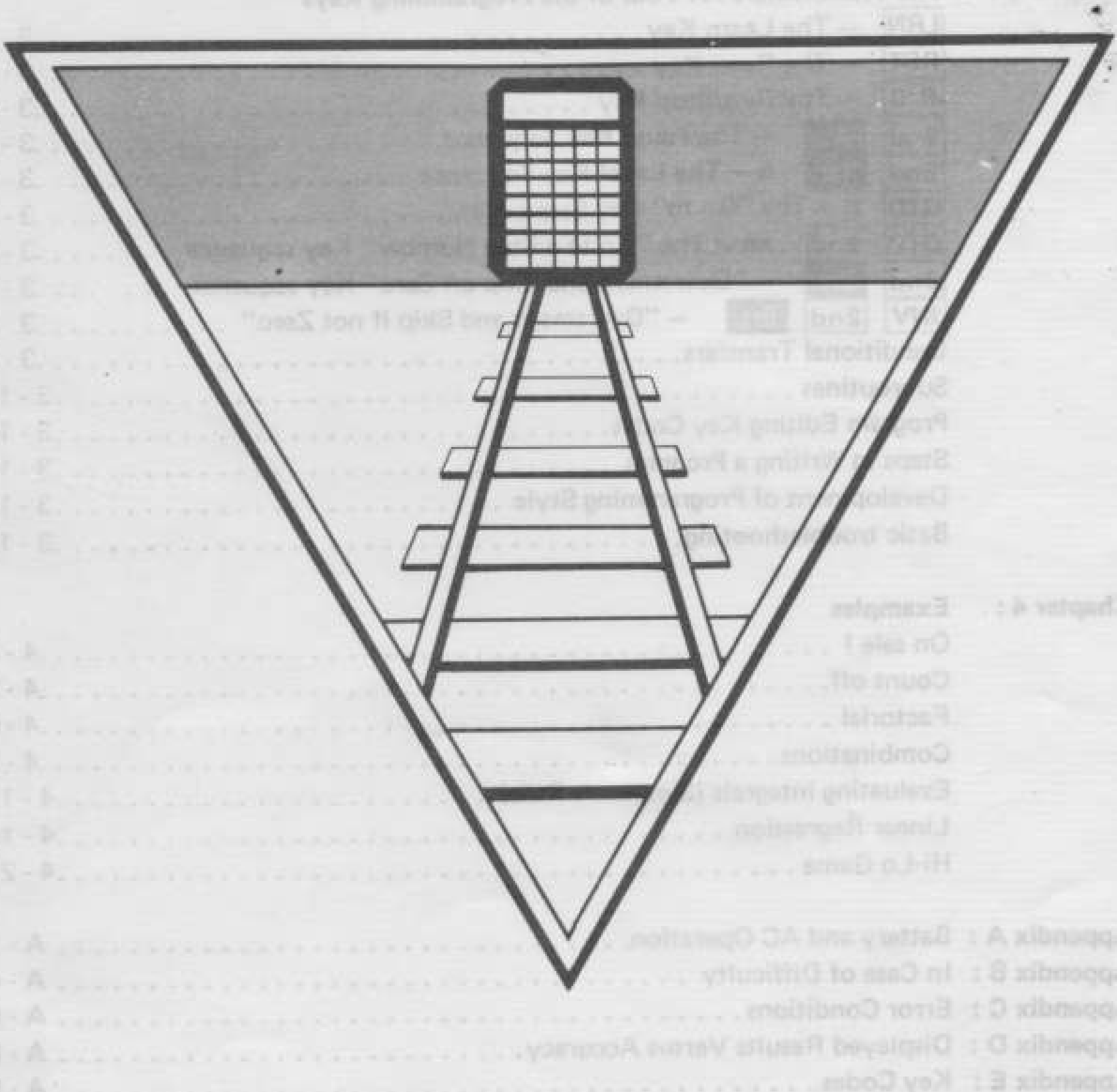


TABLE OF CONTENTS

Key Index . . . inside front cover

Chapter 1 : **Getting on Track**

A First Look at the TI-Programmable 57 . . . 1 - 1
How Does It Work ? . . . 1 - 2
A Practical Case . . . 1 - 6

Chapter 2 : **Firing Up ! A Tour of the Keys and Functions**

Section 1. Basic Keys and Functions . . . 2 - 2
Section 2. Advanced "Slide Rule" Keys . . . 2 - 10
Section 3. Statistical Functions and Keys . . . 2 - 22

Chapter 3 : **The Trackmakers : A Tour of the Programming Keys**

LRN — The Learn Key . . . 3 - 2
RST — The Resel Key . . . 3 - 2
R/S — The Run/Stop Key . . . 3 - 3
2nd Pause — The Pause Key Sequence. . . . 3 - 3
2nd Lbl n — The Label Key Sequence . . . 3 - 4
GTO n — The "Go to" Key Sequence . . . 3 - 5
GTO 2nd nn — The "Go to a Step Number" Key sequence . . . 3 - 5
2nd Dsz — "Decrement and Skip on Zero" Key sequence. . . . 3 - 6
INV 2nd Dsz — "Decrement and Skip if not Zero" . . . 3 - 7
Conditional Transfers . . . 3 - 8
Subroutines . . . 3 - 11
Program Editing Key Codes . . . 3 - 13
Steps in Writing a Program . . . 3 - 16
Development of Programming Style . . . 3 - 17
Basic troubleshooting . . . 3 - 18

Chapter 4 : **Examples**

On sale ! . . . 4 - 1
Count off . . . 4 - 3
Factorial . . . 4 - 6
Combinations . . . 4 - 9
Evaluating integrals (Simpson's Rule) . . . 4 - 11
Linear Regression . . . 4 - 15
Hi-Lo Game . . . 4 - 21

Appendix A : Battery and AC Operation . . . A - 1
Appendix B : In Case of Difficulty . . . A - 4
Appendix C : Error Conditions . . . A - 6
Appendix D : Displayed Results Versus Accuracy . . . A - 8
Appendix E : Key Codes . . . A - 9
Typical answers to "next stop" problems . . . A - 10

A FIRST LOOK AT THE TI PROGRAMMABLE 57

The calculator you're about to begin using is really straight out of the "world of science fiction" of just a few months ago. Only the most recent advancements in solid-state technology have made it possible to put such an easy and fun-to-use bundle of power right in the palm of your hand.

Easy to Use

A quick look at your calculator will tell you that it's packed with lots of keys and features. Introducing you, step-by-step, to each of these keys and features is one primary purpose of this book. Above and beyond all these features, however, is one overriding characteristic that was designed into your *TI Programmable 57* right from the start: *It's easy to use.* It's easy to make it do many things that can be enormously helpful in your schoolwork, extremely powerful in your career, and a lot of fun in your everyday life. It's designed to be a working tool that, working together with the information contained in this book, becomes a system for problem-solving and a key to discovery. We hope you'll enjoy using and exploring with it.

Starting at the Top

In learning about your machine, let's start right off with the nameplate — and a brief look at why we called it what we did:

TI — The calculator is built by *Texas Instruments Incorporated.*

Programmable — As you will be seeing in a few seconds, it is easy to "teach" your machine to help in solving problems, evaluating formulas and building "models" of your world. You can lay down a set of "Tracks" that the machine will follow as it solves a problem. A word, recognizable to some folks, that describes such a machine is already around: "Programmable". This machine, however, is a new thing — with a whole new, easy approach. So if "programmable" is a word that scares you — don't let it. Using your "57" is a cinch.

57 — This machine is the latest in a growing family of Texas Instruments programmable calculators, and it's "following in the footsteps" of another machine, our *SR-56*. The *57* is an even easier machine to use — designed for especially straightforward use.

How Does It Work?

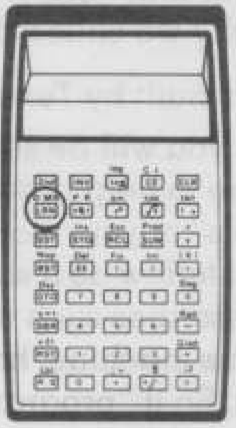
Simple! For starters, take out the calculator and check out the "heft" of this little device. Light, isn't it? That's because most of the important things inside this calculator are handled with a device that the folks at TI call a "SOAP" bar.

Now, turn it on! A zero in the display tells you that you're ready for action. (If the zero doesn't come on, or the display seems to be doing "funny" things, don't worry. The battery just needs charging. Turn the machine off, plug in your charger and wait a few minutes; then you'll be set to go.)

Let's get right into it with a simple example.
Imagine this situation:
You've just won a rather weird award from a major radio station. You'll be given 2¢! This money will be doubled every day for three weeks (21 days). *OR* – you can have $5000 in cash immediately. You have two minutes to make a decision – the announcer is waiting. (This is actually an "old and classic" problem. Watch how your calculator helps.)

Basically, you can have your machine read out for you just how your money will "grow". Here's how:

Over on the upper left hand corner of your machine, one row down from the top, is a key that is the real "jewel": it's labelled LRN



This is the "learn" key, and when you press it one time, your calculator begins to learn and remember whatever you put in next. When you press LRN again, you're telling the calculator that you've "taught" it everything you want it to know for now. The calculator stops learning at this point, but remembers all the steps you taught it. (The LRN key is like an "ON/OFF" button for teaching your calculator to do things.)

So press LRN. (At this point your display changes to **00 00**.) Now, if you decide to take the "2¢ option" on the contest, you'll be *doubling* your money for 21 days, and you want to watch it grow. To do this, Press: × 2 =.

Now another "little jewel" – right above the key labelled SST you'll see the word "Pause" written on the face of the machine. Whenever a word or function is printed above a key – on the face of the machine instead of right on the key – that's called a "second function".

Second functions allow us to pack all the power we can into your calculator, without loading it with keys. To use a second function, just push the 2nd key – in the upper left hand corner, and *then* the key right below the second function. In this book, we will use keys with a black background ■ to indicate second functions. So:
PRESS: 2nd Pause (You'll press 2nd and the key labelled "SST".)

The Pause key tells your machine:
"Stop for a moment and let's see what's going on".

One more thing – we want to *keep* doubling our money again and again. To let your machine know this, just find the RST (reset) key (close to the lower left corner). This key says, "go back and do it again, from the top". PRESS: RST

You have taught your machine all it needs to know at this point, so press LRN again, and press RST again to be sure everything is back at the beginning.

Now, to watch your 2¢ "grow" – first:
Enter 2¢ on your calculator: PRESS . 0 2 (The display should read **0.02**.)

At this point, your calculator is ready to "grow your money" – each display readout will be double the value before it. You'll start things going and count the days as they go by. You'll have 0.04 for the first day, etc. When you reach 21 days, the cash value in the display is what you will win.

O.K. – to start the machine, just press the RUN/STOP key – R/S – the lower left key on the board. Keep your eye on the display, count out 21 days, then press the R/S key again and hold it down for a moment.

On day 21 you'll have won $41,943.04! Tell the announcer you'll take the "2¢ option".

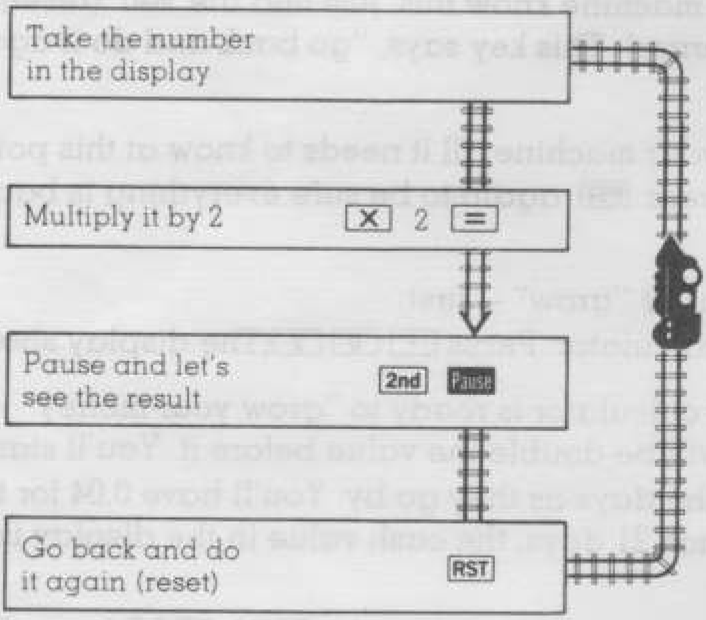
CONGRATULATIONS!

In this little example, you have already covered many of the key points that make your programmable calculator easy and fun to use:

- The LRN key lets you "teach the calculator". (You can teach it up to 50 steps, and there is a lot it can learn – any of those "steps" is usually equivalent to any keystroke, but some steps can involve two or more keystrokes.)
- The RST key starts things back at the beginning.
- The R/S key allows you to control the action – stopping and starting where you please.

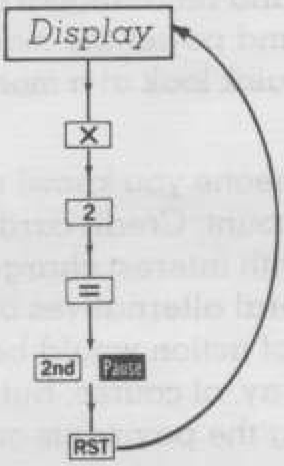
MAKING TRACKS

One way of looking at how the calculator works is to think of it as a little "railroad" – where you can "make tracks" for the machine to follow:



The "train" on this track has a special name – the "Pointer" (or, more technically, the program pointer). Each time this train passes an operation, the calculator follows the instruction or performs the operation listed. The calculator in effect *pushes its own buttons* for you – in the exact sequence you tell it. When you push the RST key (or the train comes to RST as part of a program), the train immediately goes back to the beginning – to the first step (step number 00), and starts over. To get the "train" moving, or to stop it once it's moving, you just use the R/S ("run/stop") key.

In the example you considered earlier, when you press R/S, the machine simply pressed its own buttons and doubled the display value:



and would have kept on doing this for quite awhile – had you not stopped things after 21 "go-rounds" with the R/S key. (As you'll see later on, these repetitive "go-round" situations in your calculator – and in larger computers – are called "loops".)

At this point you're already well into many of the features that make your programmable calculator so special. Actually, we've explained enough here so that you can now explore all sorts of programs on your own. We'll naturally show you all the ins and outs of your machine with more on how to solve problems and run programs using it, later in the manual. But don't be afraid to explore. There's really nothing to it, and you can't hurt the calculator by pressing any combination of buttons. Your calculator can be easily programmed to run through any set of keystrokes (up to 50 steps) which you could handle manually from the keyboard.

Just: Push LRN

- Enter your keystrokes
  (You'll need to use RST as your last step if you want things to repeat, or R/S as your last step if you want things to stop after one time. Also, if you need to enter a number midway in a program, or want to see some intermediate result, just put an R/S at that point.)
- Press LRN again
- Press RST

At this point you've programmed your calculator. To run the program, just:

- Enter the number(s) you want to work on
- Press R/S

A PRACTICAL CASE

We'll be showing you many ways to use your programmable calculator for a variety of school, home and recreational applications. You'll see with each example the ease and power the machine can bring to everyday mathematics. Let's take a quick look at a more practical case.

In this situation let's say that you (or someone you know) ran up a rather large bill (say $500) on a credit card account. Credit cards are convenient, but they really "sock you" with interest charges – typically 1.5% per month. You'd like to weigh several alternatives on payments and get a picture as to what the best course of action would be. Paying back the cash right away would be the best way, of course, but you might not be able to afford it. However, if you string the payments out over a long period, it would cost quite a bit.

Your calculator can help by speedily letting you calculate the alternatives.

For right now, let's say you'd like to know what your monthly payments would be if you paid off in 3, 6, 9, or 12 months, and how much you'd be paying in interest in each case.

HERE'S ALL YOU DO:

First, turn your calculator off and on – to clear everything – and then push the following keys. (Remember, we'll be covering all the details later – for now just see what your calculator will do for you! Push each key carefully.)

LRN
RCL 1 ×
(RCL 2 ÷
(1 − (1 + RCL 2)
y^x RCL 3 +/−)) =
R/S
× RCL 3 =
− RCL 1 =
R/S (At this point your display should read **27 00**. If it doesn't, turn your calculator OFF and ON, and go back and re-key carefully.)

Now press:

LRN

At this point, all you need to do is store the values you want to try out in the three memories we've used, and let your calculator do the rest.

Enter the amount you owe and store it in memory one by pressing:

500 STO **1**

Enter your monthly interest rate (i = 1.5% = .015) and store it in memory two.

.015 STO **2**

Enter the first option you'd like to try – equal payments for n = 3 months:

3 STO **3**

From now on in it's a cinch!
First press 2nd fix **2** so we'll be reading out dollars and cents clearly in the calculators display.
Then press RST R/S.

There it is: If you decided to pay back in 3 months, your monthly payment would be – $171.69.

Press R/S again to see how much you'd be charged for interest – $15.07.

To try the other options:

Enter 6 months, press 6 STO **3**
Then press RST R/S
Your payment: $87.76
Press R/S again for interest charges: $26.58

Enter 9 months, press 9 STO **3**
Then press RST R/S
Your payment: $59.80
Press R/S again for interest charges: $38.24

Enter 12 months, press 12 STO **3**
Then press RST R/S
Your payment: $45.84
Press R/S again for interest charges: $50.08

You can now go on and pick a payment program very carefully, seeing exactly how much your interest charge will be for each alternative.

In this example you've used your calculator to handle a very important and versatile business formula. The formula for calculating the amount of (equal monthly) payments on a credit card type of account is:

$$\text{Payment} = \text{PV} \times \left(\frac{i}{1 - (1 + i)^{-n}} \right)$$

Where: i = the *monthly* interest rate (1.5% or .015)
n = number of months
PV = the *present value* of the balance you owe ($500)

The *total interest charge* in dollars is calculated as:
Total Interest Charge = (n × payment) − PV
In handling the problem above the present value of the balance owed (PV) was stored in memory 1.
The monthly,interest rate (i%) was stored in memory 2, and the number of months you take (n) was stored in memory 3.

From now on in you can "try out" any number of months you'd like (for any present value, or any interest rate) just by changing what you've got stored in one of the three memories. Your calculator does all the work at the touch of a key.

Moving On

Well, at this point you've already come quite a long way. You've seen your calculator in action in two program situations – each of which has a useful side as well as a fun side. What we hope is that you've also seen how easy it is to use and program your calculator. Let's make a few major points about your machine before we move on:

First of all, your programmable calculator is versatile and powerful – and you can do all sorts of manual calculations right off the keyboard anytime.

Secondly, your calculator is programmable – and all this really means is that it can remember keystroke instructions you give it, then go back and push its own buttons for you. We've seen that programming the machine to follow instructions is quite easy.

- Just push LRN
- Enter your keystroke instructions.
- Push LRN again
- Push RST (don't forget this one – your program needs to start at the right place, and RST starts it back at the first step.)
- From then on in, you just enter numbers you want to work on and press R/S.

This new dimension of programmability brings several special benefits to you. Faster and more accurate calculations – particularly in repetitive situations – are now possible. You'll find it easier to use your calculator as part of decision-making processes, in exploring relationships in math, and in a host of everyday life and career applications.

Thirdly, and probably most important, your programmable calculator should be a lot of fun to use! Don't be afraid to explore with it.

A TOUR OF KEYS AND FUNCTIONS

Your calculator is a powerful problem-solving device that's especially designed for easy use. You've already seen some simple programs in action, and we'll continue to discuss the programming side of your calculator's capability in later chapters. But apart from its programmability – your calculator is a versatile and powerful "slide rule" calculator. It's always ready to handle calculations right from the keyboard. In this chapter we'll be touring the direct problem-solving keys on your machine – the keys you can use immediately to get answers. Then, in later chapters, we'll show you how these basic keys can be used with the others on the keyboard to create programs.

The keyboard of your calculator has been organized and arranged in a common sense, straightforward way.

You can easily do simple things like balancing your checkbook or adding your grocery bill, as well as complex technical problems. The usefulness of any machine or tool, however, depends on the person who operates it. You'll want to get familiar with *all* of its features – all of what it will (and will not) do for you.

To get full use from your calculator, take the few minutes necessary just to see each key in action!

To make it easy for you to get acquainted, this tour is divided into three major sections:

Basic Keys and Functions
Advanced/"Slide Rule" Keys & Functions
Statistical Functions & Keys

For those of you who have already owned or are familiar with an advanced slide rule calculator – you may want to skip to *Chapter 3* and get right into programming. If there are some keys on the machine you don't know about, however, we'd advise that you take this quick tour. Take out your calculator and keep it handy as we go through the keys – relax – and let's go.

SECTION 1: BASIC KEYS AND FUNCTIONS

We'll begin with a brief look at the basic "chassis" on your calculator, before we go on to check out the "extra options". These basics are what allow you to get information into and out of the machine – and let you handle the arithmetic part of mathematics quickly and accurately.

THE DISPLAY

Whenever you first turn on your calculator, you should see a single zero in the display indicating that all is well, the machine's on, and it's ready for action. Just turning the calculator OFF and ON clears everything inside. To check out your calculator's display, press the [8] key, the decimal point key [•], the change sign key [+/-], and then push the [8] key until the whole display is lit up. You can enter up to 8 digits into your calculator at any one time, for both positive and negative numbers. (Entries after the 8th digit are ignored.) For extra accuracy, however, results of calculations are computed to eleven digits inside your calculator, and then rounded off to 8 digits in the display. Notice that the negative sign stays immediately to the left of any negative number in the display, for easy reading.

To continue the tour press the clear key [CLR] in the upper right corner of keyboard, and read on.

[2nd] AND [INV]: THE "DUAL FUNCTION" KEYS

Your calculator is loaded with functions to make all sorts of calculations easy and accurate. To allow you access to all of this power, without loading the machine with keys, many of the calculator keys have more than one function. The first function of the key is printed right on it. To use the first function on any key – just press it. The second function of a key is printed right above it. To use second functions, just press the [2nd] key, (upper left on the keyboard), and then the key right below the function you wish to use. (We'll indicate second functions in this book with a black background key ■. For example, to put a π in the display you'll use the key sequence [2nd] [π].)

The inverse key – [INV] – also provides additional calculator functions without increasing the number of keys on the keyboard. The [INV] key "reverses" the purpose or function of certain keys. *Note:* In cases where you need to use both the [2nd] and [INV] keys – you can use them in either order and get the same result. If you use the [INV] before a function with *no* inverse, it's simply ignored.

CLEARING THE CALCULATOR

There are several procedures that allow you to clear various parts of your calculator, or to clear the entire machine – depending on your needs as you proceed through a problem.

CE The CE (clear entry) key clears the last number you entered into the calculator, as long as that number wasn't followed by a function or operation key. (So if you accidentally hit a 5 instead of a 6 in the middle of an entry, just hit CE and try again.) This key will also stop the display from flashing if you've created an error condition in your calculator (we'll say more about this later). The CE key, however, doesn't affect pending operations, what's stored in the memories, or calculated results.

CLR–The CLR (clear) key (upper right on your machine) essentially clears the entire machine, *except* for statistical data stored in the memories, settings made on the display format, and program steps.

2nd **C.t** – This allows you to clear only the "t" register (or "t" memory). This capability is beneficial in programming and with statistical functions. We'll look at it in more detail later.

INV **2nd** **C.t** – This key sequence clears everything in the machine except program steps and decimal settings. (Technically, this doesn't clear the first two registers of the math stack, but this will seldom be a problem.)

OFF-ON – Turning the calculator OFF, then ON again clears everything.

In fact, it's one sure way to quickly clear all your program steps, memories, display settings, everything!

0 - 9 • +/− – Data Entry Keys

Your calculator operates with a full floating decimal point, and numbers are entered into the machine with the data entry keys 0 - 9 • +/−. As you enter any number, the decimal point stays to the right of your entry until the decimal point key (•) is pressed. After pressing the decimal key, the fractional part of the number is keyed in, and the decimal point floats to the left with it. To change the sign of a number in the display, just push the change sign key +/− once. (Pressing +/− again changes the sign back again.)

+ − × ÷ and = – Basic Operation Keys

Basic arithmetic is handled with the 5 basic operation keys: + − × ÷ and =. Your calculator is equipped with a simple and powerful entry system – the AOS™ entry method. This entry system makes problem solution exceptionally easy. You just key in the problem the way it's written, press =, and get your result. The amazing feature of the AOS system is that it automatically sorts out mixed operations in a problem for you, and applies them in the correct order as it calculates your result. (We'll say more about the AOS entry system in the next section.)

When you press the = key, all pending operations (things waiting to happen inside your calculator) are completed. You get your result, and the calculator is cleared – ready to start on the next problem.

Example: Calculate 15 + 7 × 31 − 4 = ?

Press	**Display/Comments**
15 + 7 × 31	
− 4 =	**228.**

Note: The AOS system of entry makes it easy to get the right answer in this example, and not all calculators have it.

THE AOS™ ENTRY METHOD

Mathematics is a science which adheres to a variety of rules. One such rule is that it never permits two different answers to the same series of operations. Because of this requirement – one solution for any computation – mathematicians have established a set of universally accepted rules when mixed operations are used in one calculation. For example, the problem:

$$3 + 10 - 2 \times 14 \div 7 = ?$$

has *only one right answer!* (Know what it is? It's 9.)

You can key this problem directly, left to right, into your calculator equipped with the AOS entry system and you'll get the correct answer. The calculator sorts the operations you enter, applies them in the correct order, and lets you see what it's doing along the way. Your calculator's AOS entry system is quite an organizer! It sorts and then performs operations it receives from you in the following universally accepted order:

1) Special Single Variable function keys ($\boxed{x^2}$ $\boxed{\sqrt{x}}$ etc.) – act on the displayed number immediately – as soon as you push the key. (We'll talk more about each of these keys later in the "tour" – but they include all the keys for the trig and log functions and their inverses, as well as square root, and reciprocal keys.)
2) Exponential calculations $\boxed{y^x}$ and $\boxed{\text{INV}}$ $\boxed{y^x}$ (or $\sqrt[x]{y}$) are done next (we'll discuss these further in a following section).
3) Multiplications and divisions are completed next, in order from left to right, followed by
4) Additions and subtractions, in order from left to right.

Finally, the equals key $\boxed{=}$ completes all operations.

When you were in elementary school you may have heard the memory aid "My Dear Aunt Sally" (MDAS) applied to help you remember the last part of this hierarchy: Multiplications and Divisions first, in order left to right – then Additions and Subtractions in the same way. In a calculator equipped with the AOS entry system – all of this is remembered for you.

There are cases in problem-solving where *you* may want to exactly specify the order in which an expression is evaluated, or the way in which a problem is completed. In these cases you can control the order with the parentheses keys $\boxed{(}$ $\boxed{)}$ discussed in the next section. Parentheses demand a special first level of attention in mathematics – and they're treated that way by your calculator.

$\boxed{(}$ $\boxed{)}$ – PARENTHESES KEYS

In a variety of problems you may need to specify the exact order in which expressions are evaluated, or the way in which numbers are grouped, as a problem is solved. Parentheses give you a way to "cluster" numbers and operations. By putting a series of numbers and operations in parentheses you tell the calculator: "Evaluate this little problem first – down to a single number result, then use this result for the rest of the calculation." *Within* each set of parentheses your calculator will operate according to the rules of algebraic hierarchy. You should use parentheses if you have any doubts in your mind about how the calculator will handle an expression.

There is a limit to the number of parentheses that can be opened at one time, and how many "pending" operations can be handled. Your calculator allows you to open nine parentheses at one time, with up to four operations pending – exceeding these limits results in a flashing display. (You'll rarely encounter this as a problem.)

Note an important point when using parentheses. You may often see equations or expressions written with parentheses to indicate implied multiplication: $(2+1)\ (3+2)=15$. Your calculator will *not* recognize implied multiplications. You must key in the operation between the parentheses:

$\boxed{(}\,2\,\boxed{+}\,1\,\boxed{)}\,\boxed{\times}\,\boxed{(}\,3\,\boxed{+}\,2\,\boxed{)}\,\boxed{=}\,15.$

Here's an example on using parentheses:

$$\text{Evaluate}\quad \frac{(8\times 4)\ +\ (9\times -\ 19)}{(3+10\div 7)\times\ 2}=$$

Solution: In problems of this type – you want the calculator to evaluate the entire numerator, then divide by the entire denominator. You can be sure of this taking place by placing an extra set of parentheses around the numerator and denominator as you key the problem in.

PRESS	**DISPLAY/COMMENTS**
$\boxed{(}\,\boxed{(}\,8\,\boxed{\times}\,4\,\boxed{)}\,\boxed{+}$	**32.** (8x4) displayed
$\boxed{(}\,9\,\boxed{\times}\,19\,\boxed{+/-}\,\boxed{)}$	
$\boxed{)}\,\boxed{\div}$	**–139.** The value of the numerator
$\boxed{(}\,\boxed{(}\,3\,\boxed{+}\,10\,\boxed{\div}\,7$	
$\boxed{)}\,\boxed{\times}\,2\,\boxed{)}$	**8.8571429** Value of denominator
$\boxed{=}$	**–15.693548** The final result.

MEMORY KEYS

There are 8 multi-purpose memories available for you to use in your calculator. These memories are special locations in the machine where you can store numbers you may need to use later on. The memories are a real bonus – and in many ways give you "several calculators" in one, since you can store, recall or perform arithmetic on the numbers in the memories without affecting calculations you have in progress in the "main machine".

The [CE] and [CLR] keys will not affect what's in the memories – but you can use the [INV] [2nd] [C.I.] key sequence to clear them all out if you need to. (Turning the calculator OFF and ON does this, too.)

Since you have 8 memories you need to tell the machine which one you want to work with at any given time. Every time you push a memory key you need to follow it immediately with the number of the memory you are using (**n** = **0, 1, 2, 3, 4, 5, 6,** or **7**). This tells the machine which one of the 8 memories you're referring to at the moment. The operations of the memory keys is pretty much "common sense" as shown below.

[STO] n – THE STORE KEY

This key just "stores" the displayed number in the memory you specify with **n.** (**n** = **0, 1, 2, . . . 7**). (Any number previously stored in memory **n** is automatically cleared out first.)

[RCL] n – THE RECALL KEY

Any time you press [RCL] **n**, the number stored in memory **n** appears in the display and can be used in operations and calculations. The number remains in the memory after you press [RCL] **n**, and you can recall the value from any memory as many times as you need to in any calculation. A stored number remains in memory until you alter it with another memory operation, the [INV] [2nd] [C.I.] clearing operation, or by turning the calculator off.

An example on the use of the memories:

Let's say that:
$a = 10.3\ (25 - 1.7)$
$b = 15a + 6$
$c = 20b$, and you need to find c.

PRESS	DISPLAY/COMMENTS
10.3 [×] [(] 25 [−]	Calculate and store a
1.7 [)] [=]	**239.99**
[STO] **1**	**239.99** a stored in memory 1
15 [×] [RCL] **1** [+] 6 [=]	**3605.85**
[STO] **2**	**3605.85** b stored in memory 2
20 [×] [RCL] **2** [=]	**72117.** The value of c.

MEMORY ARITHMETIC

In addition to the basic memory keys, there are a series of keys that let you perform arithmetic on the numbers stored in memory without affecting other calculations in progress:

[SUM] **n** – The Sum Key – allows you to algebraically add whatever number is in the display directly to the number stored in any memory. (This doesn't affect any calculations in progress.) The result of the addition stays stored in the memory. *Note:* This process is different from what happens when you use the [STO] key. The [STO] **n** operation clears out the number in the memory, and replaces it with the number in the display. (The display is not affected by this operation.)

[INV] [SUM] **n** – The Subtract key sequence, subtracts the number in the display from the number in memory **n**. The result stays stored in memory **n**. (The display does not change.)

[2nd] [Prd] **n** – The "Multiply" or Product key sequence, multiplies the number in memory **n** by what's in the display. The product stays stored in memory **n**. (The display does not change.)

[INV] [2nd] [Prd] **n** – The "Divide" or Quotient key sequence, divides the number in memory **n** by the number in the display. The quotient stays stored in memory **n**. (Note the [INV] and [2nd] may be pressed in either order for this sequence: [2nd] [INV] [Prd] **n** for any "memory divide" calculation.) The display does not change during this operation.

[2nd] [Exc] **n** – The Exchange key sequence just "swaps" the number in memory **n** with the number in the display. (The display value gets stored in memory **n**, while the number stored in memory **n** is displayed.)

[x:t] – **The "x" Exchange with "t" Key**

This is a special exchange key that exchanges the number in the display with the number in memory **7**. Memory **7** is used in several special functions on your machine and is given a special name: the "t" or "test" register. The [x:t] key is identical in function to the [2nd] [Exc] **7** key sequence.

Additional Notes on Memories:

The 8 memories in your machine are designed primarily for your use and convenience – but there are times (particularly in more complex calculations) where your machine needs extra "space" to work in. In these cases your calculator will need to use the memories, and this can affect numbers you have stored in them. We'll mention these cases here – and review them again as various special machine features are discussed.

- When you're evaluating a complex expression, involving 3 or 4 levels of pending operations, memories **5** and **6** will be used.
- Memory **7**, the t register, will be used whenever the [x:t] key is used.
- Statistical functions and trend line analysis problems (discussed later) may use memories **0, 1, 2, 3, 4, 5** and **7.**
- The [2nd] [Dsz] key sequence, which is discussed in *Chapter 3*, works with memory zero.

In practice these memory situations aren't often a problem – but if you're really "loading up" the machine with work to do, you need to be aware of the fact that some of what you may have stored in memories can be affected.

SECTION 2: ADVANCED/ "SLIDE RULE" KEYS

In this section we move on to discuss some of the features of your calculator that are especially helpful in engineering, scientific, and more advanced mathematical applications (applications that not too long ago used to be handled with mechanical slide rules and tables). Many of these features have only been made possible by more recent developments in Integrated Circuit (IC) Technology. These new developments allow calculator designers to "put" various functions and tables right into the IC chip. This way you can put the calculator to work on a complex calculation with the touch of a key.

EE – SCIENTIFIC NOTATION KEY

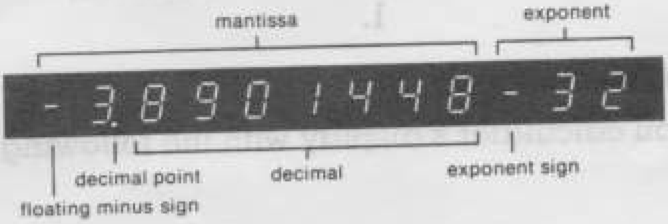
In many applications, particularly in science and engineering, you may need to calculate with extremely large or small numbers. Such numbers are easily handled (by both you and your calculator) using scientific notation. A number in scientific notation is expressed as a base number (or "mantissa") times ten raised to some power (or "exponent").

$$\text{Mantissa} \times 10^{\text{power}}$$

To enter a number in scientific notation

- Enter the mantissa (then press +/- if it's negative.)
- Press EE (Enter Exponent) – a "00" will appear at the right of the display.
- Enter the power of 10 (then press +/- if it's negative.)

A number such as $-3.8901448 \times 10^{-32}$ will look like this in your display:



In scientific notation the power of ten (the two digits to the right in your display) tells you where the decimal point would have to be if you were writing the number out in longhand. A positive exponent tells you how many places the decimal point should be shifted to the right, a negative exponent – how many places to the left. For example:

2.9979×10^{8} equals $2\underbrace{.99790000}$
(Move decimal 8 places right, add zeroes as needed.)

1.6021×10^{-19} equals $\underbrace{.000\,000\,000\,000\,000\,000\,1}6021$ (Move decimal 19 places left, add zeroes as needed.)

2nd Fix n – FIX DECIMAL CONTROL

This very convenient feature allows you to choose the number of digits you'd like to have displayed to the right of the decimal point as you go through your calculations. Just press 2nd Fix then press the desired number of decimal places (0 to 8). Try it – press 2nd Fix 3 – the display immediately changes to "0.000".

The calculator will round all of your subsequent results to this number of decimal places. You can go on and make number entries with as many digits as you like, and the calculator will retain its own internal (11-digit) accuracy. The display value will continue to be correctly rounded to the number of decimal places you've selected. Note also that you can use the fix key to set the desired number of decimal places whether you're in standard display format or scientific notation. Turning your calculator OFF and ON again erases the "fix" condition. Try the following example.

Example: ⅔ = 0.6666667

PRESS	**DISPLAY/COMMENTS**	
OFF-ON	0	
CLR	0	
2 ÷ 3 =	**0.6666667**	
2nd Fix 6	**0.666667**	(*Note:* display value is correctly rounded)
2nd Fix 2	**0.67**	
2nd Fix 1	**0.7**	
2nd Fix 0	**1.**	

TO CLEAR THE "FIX" CONDITION:

You can "unfix" you calculator's display with the following key sequences:

INV 2nd Fix

2nd Fix 9

Turning the calculator OFF and ON (this clears everything).

2nd π – "Pi" Key Sequence

The 2nd π key sequence displays the first 8 digits of π. (Eleven digits are entered into the calculator – 8 correctly rounded digits are displayed.) The number you'll see displayed is: **3.1415927**. This key sequence displays π immediately, doesn't affect calculations in progress, and can be used anytime in a calculation.

2nd |x| – Absolute Value Key Sequence

When this key sequence is pressed, the sign of the number in the display (x) is made positive. This feature comes in handy in a variety of calculating (and programming) situations.

When 2nd |x| is pressed:
If the sign of the number in the display is negative, it's changed to positive.
If the sign of the number in the display is positive, things are left alone.
This key can be used any time in a calculation, and doesn't affect calculations in progress.

Note that several keys on the keyboard use the letter "x" as part of their marking. The "x" on these keys just means "the number in the display".

x^2 $\sqrt{x}$ 1/x – Square, Square Root and Reciprocal Keys

These 3 easily accessible keys are essentials for speedily handling a variety of algebraic and equation solving situations. All three of these keys act immediately on the number in the display (x), and don't affect calculations in progress.

x^2 – The Square Key – calculates the square of the number in the display (multiplies the displayed number by itself).

$\sqrt{x}$ – Square Root Key – Calculates the square root of the number in the display. The square root of a number (say x) is another number (labelled $\sqrt{x}$), such that:

$(\sqrt{x}) \times (\sqrt{x}) = x.$

1/x – The Reciprocal Key – Divides the displayed number into one.

y^x – Universal Power Key

This powerful key allows you to raise any (positive) number to any power at the touch of a key. To use this key:

- Enter the number you want to raise to a power (y)
- Press y^x
- Enter the power (x)
- Press = (or any operation key)

Example: Calculate $3.1897^{4.7343}$

Press	**Display/Comments**
CLR	**0** Clear
3.1897 y^x	**3.1897** "y" value
4.7343	**4.7343** "x" value
=	**242.60674** Final result: y^x

INV y^x –Universal Root Key Sequence ($\sqrt[x]{y}$)

This key sequence allows you to take any root of any positive number (the inverse of y^x- or the $\sqrt[x]{y}$). Before calculators came along, calculations like this were usually pretty time consuming – and involved a set of logarithm tables.) To use this key sequence to take $\sqrt[x]{y}$:

- Enter the number you want to take the root of (y)
- Press INV y^x
- Enter the root you want to take (x)
- Press = (or any operation key)

Example: Calculate $\sqrt[3.871]{21.496}$

Press	**Display/Comments**
CLR	**0** Clear
21.496 INV y^x	**21.496** "y" value
3.871	**3.871** "x" value
=	**2.2089685** Final result ($\sqrt[x]{y}$).

Note: The y^x key (and INV y^x) key sequences perform calculations that involve quite a few steps inside your calculator, and it takes a little longer to complete these calculations than others. *Be sure you wait for your calculator to finish – with a final result in the display before pressing additional keys.*

lnx AND 2nd log – Logarithm Keys

Logarithms are mathematical functions that enter into a variety of technical and theoretical calculations. In addition, they form an important part of many mathematical "models" of natural phenomena. The logarithm keys give you immediate access to the "log" of any number – without having to hassle with bulky tables.

lnx – The Natural Logarithm Key – immediately displays the natural logarithms (base e = 2.7182818) of the number in the display. (*Note:* the number in the display must be positive – attempting to take lnx of a negative number will result in a flashing display.)

2nd log – The Common Logarithm Key – immediately displays the common logarithm (base 10) of the (positive) number in the display.

INV lnx AND INV 2nd log – Anti-Logarithm Key Sequences

These key sequences are the "inverse" or "anti" functions of the logarithms. ($e^{(\ln x)} = x$, and $10^{(\log x)} = x$.) These calculations arise in many technical situations and can be handled quickly with just a few keystrokes on your calculator.

INV lnx – e^x Key Sequence – Raises e to the power of the number in the display (calculates the natural antilogarithm of the number in the display).

INV 2nd log – 10^x Key Sequence – Raises 10 to the power of the number in the display (calculates the common antilogarithm of the display value).

Notes on logarithm and "anti" logarithm keys:

> Each of these keys acts immediately on the number in the display – and doesn't affect calculations in progress. Your calculator uses a variety of "routines" for arriving at these values. Certain limits are set by these routines – and exeeding them may result in an error indication · see the *Appendix* C on error conditions for details.

Some examples: Calculate log 15.32, ln 203.451, $e^{-.69315}$, 10^{π}

Press	**Display/Comments**
15.32 2nd log	**1.1852588**
203.451 lnx	**5.3154252**
.69315 +/– INV lnx	**0.4999986**
2nd π 2nd INV log	**1385.4557**

2nd Int — "INTEGER PART OF A NUMBER" KEY SEQUENCE

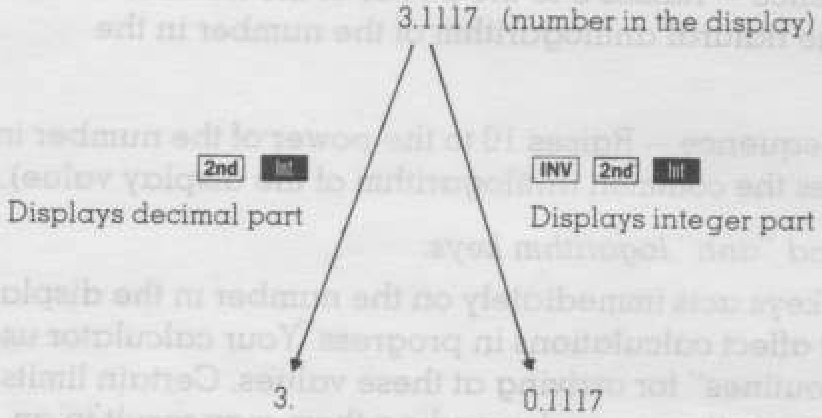
This key sequence displays the *integer part* of any number in the display. For example, if you enter the number 3.1117 into the display, and press the 2nd Int key sequence, the calculator "chops off" everything to the right of the decimal point and displays only the integer 3. (Try it!)

There are many situations in mathematics where you'll find yourself needing to deal with *just* the integer part of a number. In programming you'll find this key sequence very handy for handling these situations.

2nd INV Int —"FRACTIONAL PART OF A NUMBER" KEY SEQUENCE

There are some instances in math and programming where you need to deal only with the decimal part of a number, and need to isolate it. This key sequence displays the fractional or decimal part of any number in the display. So if you enter 3.1117, and press the 2nd INV Int key sequence, 0.1117 will be displayed.

The following diagram summarizes these two key sequences:

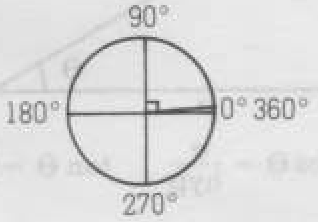


Note: The original number is lost when Int or INV Int is used — store it first if you'll need it later.

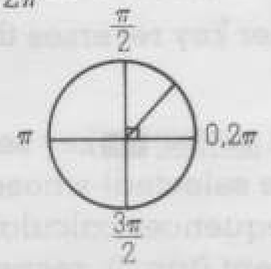
2nd Deg , 2nd Rad , 2nd Grad —ANGULAR MODE KEYS

Your calculator is equipped to handle a variety of calculations that involve angles – notably the trigonometric functions and polar/rectangular conversions. When performing these calculations, your calculator allows you to select any one of three common units for angular measure using the key sequences below:

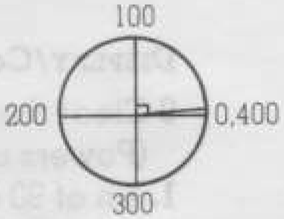
2nd Deg – selects degree mode. In this mode all entered or calculated angles are measured in degrees, until another mode is selected. (one degree equals $\frac{1}{360}$ of a circle – a right angle equals 90°)



2nd Rad – selects radian mode. In this mode all angles are measured in radians (one radian equals $\frac{1}{2\pi}$ of a circle; a right angle equals $\frac{\pi}{2}$ radians).



2nd Grad – selects grad mode. In this mode all angles are measured in grads (one grad equals $\frac{1}{400}$ of a circle – a right angle equals 100 grads).

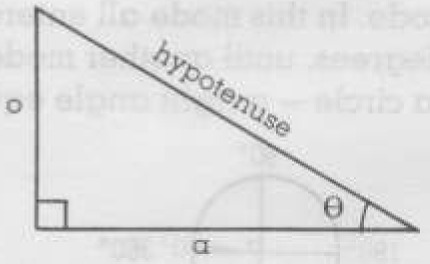


IMPORTANT NOTE: When you first turn your calculator on it powers up in the degree mode, and stays in that mode until a new mode is selected.

One very common error encountered when working with angles is finding yourself in an incorrect angular mode. Once you select an angular mode, your calculator will *stay in* that mode until you select a new one, or until you turn the calculator OFF and ON again (the calculator is always in degree mode on power up.)

2nd sin, 2nd cos, 2nd tan — TRIGONOMETRIC KEY SEQUENCES

These key sequences immediately calculate the sine, cosine, and tangent of the angle in the display (angle is measured in units of selected angle mode). The trig functions relate the angles and sides of a right triangle as shown below:



$\sin\Theta = \frac{o}{hyp}$ $\cos\Theta = \frac{a}{hyp}$ $\tan\Theta = \frac{o}{a}$

INV 2nd sin, INV 2nd cos, INV 2nd tan

Using the INV preceding another key reverses the operation and intention of that key.

The INV 2nd sin, INV 2nd cos, and INV 2nd tan key sequences calculate the *angle*, (in the units of the mode selected) whose sine, cosine or tangent is in the display. (These key sequences calculate the arcsine ($\sin^{-1}$), arccosine ($\cos^{-1}$), and arctangent ($\tan^{-1}$), respectively.)

Examples: Calculate the sine of 90°, 90 radians, and 90 grads, and the arctan of 1.

PRESS	**DISPLAY/COMMENTS**
OFF-ON CLR	**0** Clear the entire machine (Powers up in degrees mode)
90 2nd sin	**1.** sin of 90 degrees
90 2nd Rad 2nd sin	**0.8939967** sin of 90 radians
90 2nd Grad 2nd sin	**0.9876883** sin of 90 grads
1 2nd Deg INV 2nd tan	**45.** The angle (in degrees) whose tangent is one.

ANGLE CONVERSIONS

You may at times find it necessary to convert angular values from one unit system to another. While there are no special conversion keys on your calculator for this purpose, the key sequences to convert angular units in the first and fourth quadrants are pretty simple and can be used without affecting the memories or calculations in progress.

1) Be sure the calculator is in the correct angular mode and enter the angle to be converted.
2) Press 2nd sin
3) Change calculator to desired angular mode.
4) Press INV 2nd sin

Example: Express 50 degrees in radians.

PRESS	**DISPLAY/COMMENTS**
2nd Deg	**0**
50 2nd sin	**0.7660444**
2nd Rad	
INV 2nd sin	**0.8726646** Radians

The angular range of these conversions is limited to the first and fourth quadrants. For larger angles you may convert directly:

- number of degrees $\times\frac{100}{90}$ = number of grads
- number of degrees $\times\frac{\pi}{180}$ = number of radians
- number of grads $\times\frac{90}{100}$ = number of degrees
- number of grads $\times\frac{\pi}{200}$ = number of radians
- number of radians $\times\frac{180}{\pi}$ = number of degrees
- number of radians $\times\frac{200}{\pi}$ = number of grads

CONVERSIONS:

Your calculator is equipped with two keys that make it especially handy in conversion situations.

2nd D.MS — DEGREES, MINUTES, SECONDS TO DECIMAL DEGREES CONVERSIONS

This key sequence converts an angle expressed in degrees minutes and seconds to decimal degrees. The INV 2nd D.MS key sequence does just the opposite — converting *from* decimal degrees *to* degrees minutes and seconds. This conversion is handy if you have to do arithmetic with angles, or with time — expressed in hours minutes and seconds.

To enter and convert an angle, just follow the format shown on the D.MS key label:

- Enter the number of degrees
- Press the decimal point key, ·
- Enter the number of minutes (2 digits — less than 60)
- Enter the number of seconds (2 digits — less than 60)
- Press 2nd D.MS

The decimal equivalent of the angle (or time) is now displayed. The reverse conversion is easy to do also:

- Enter the angle (whole number and decimal part)
- Press INV 2nd D.MS

The angle is now displayed as the number of degrees, followed by a decimal point, followed by the number of minutes and seconds.

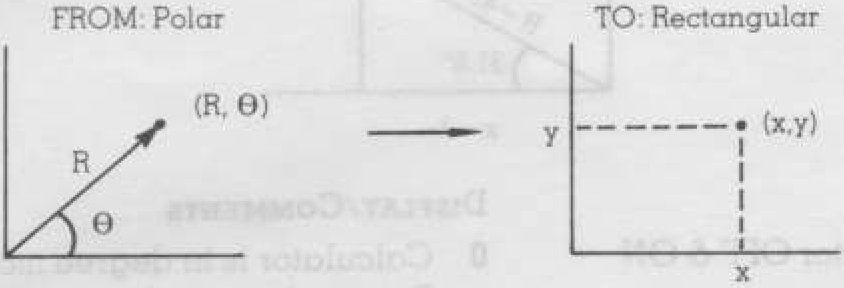
Example: Convert 47°05'38" to decimal degrees and back to degrees minutes and seconds again.

PRESS	**DISPLAY/COMMENTS**
47.0538 2nd D.MS	**47.093889** angle decimal equivalent
INV 2nd D.MS	**47.0538** back to degrees, minutes, seconds

Note that you can also convert hours, minutes, and seconds to decimal form by the same technique.

2nd P→R – Polar to Rectangular Conversions

This is a handy feature of your calculator that is particularly useful in science and engineering applications. Working with the x⇌t key – it's fast and easy to convert from polar to rectangular coordinates, or vice versa. Just follow the key sequences illustrated below:



To convert *from* polar *to* rectangular coordinates:

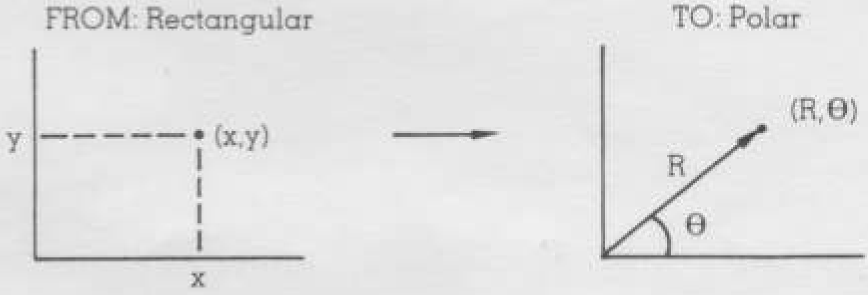
- Enter your value for "R"
- Press x⇌t
- Enter your "Θ" value (units selected with angular mode keys).
- Press 2nd P→R

"y" is now displayed; to read "x":

- Press x⇌t

"x" is now displayed.

To convert *from* rectangular *to* polar coordinates follow the sequence listed below:



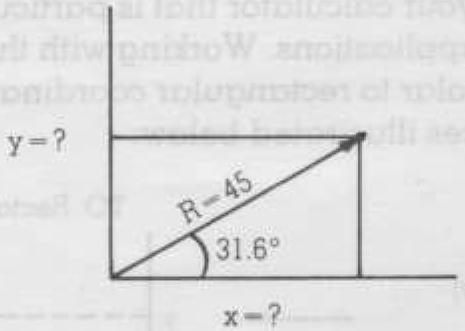
- Enter your "x" value
- Press x⇌t
- Enter your "y" value
- Press INV 2nd P→R

"Θ" is now displayed (in units selected by mode key); to read "R":

- Press x⇌t

"R" is now displayed.

Example: Convert R = 45 meters, Θ = 31.6° into rectangular coordinates



Press	**Display/Comments**
Turn calculator OFF & ON	**0** Calculator is in degree mode
45 [x:t] 31.6	Enter polar coordinates: R, then Θ.
[2nd] [P→R]	**23.579366** "y" value
[x:t]	**38.327712** "x" value

Note: This conversion will make use of memory 7 (the "t" register) — so be sure *not* to use memory 7 while you're doing a polar to rectangular conversion (you'll lose numbers stored there).

SECTION 3: STATISTICAL FUNCTIONS AND KEYS

In many situations – in school work, everyday life, or business, you may find yourself handling large sets of data points. This data could be from laboratory measurements, test scores, results from a study or research project, etc.

Your calculator is equipped with features allowing you to easily collect and analyze data by rapidly calculating mean, variance, and standard deviation. Intermediate results are stored in the memories for you in an easy to access fashion. (You'll find this especially useful in programming with statistics – as you'll see later on in this book.) You can also simultaneously analyze *two* sets of data, allowing you to examine relationships between them in advanced statistical programming.

To collect and analyze sets of data here's the procedure:

· Begin any and all statistical calculations by either turning your calculator OFF and ON, or by pressing the [INV] [2nd] [Cl] key sequence.

If you have only one set of data to analyze:

- Enter each data point
- Press [2nd] [Σ+]
- Repeat for all data points

- Press [2nd] [$\bar{x}$] to calculate the *mean* of the data
- Press [2nd] [σ^2] to calculate the *variance* of the data (with N weighting).
- Press [2nd] [σ^2] [$\sqrt{x}$] to calculate the standard deviation of the data (with N weighting). ("N weighting" means that the total number of data points is used in the calculation of the variance – this type of variance is called a population variance.)

If you have two sets of data to analyze simultaneously:

Call the two sets of data "x" (independent) and "y" (dependent) *arrays* of data.

- Enter an "x" data point
- Press [x:t]
- Enter a "y" data point
- Press [2nd] [Σ+]
- Repeat for all points

- Press [INV] [2nd] [$\bar{x}$] to calculate the mean of the "x" data points.
- Press [2nd] [$\bar{x}$] to calculate the mean of the "y" data points
- Press [INV] [2nd] [σ^2] to calculate the variance of the "x" data points
- Press [2nd] [σ^2] to calculate the variance of the "y" data points
- Press [INV] [2nd] [σ^2] [$\sqrt{x}$] to calculate the standard deviation of the "x" data points.
- Press [2nd] [σ^2] [$\sqrt{x}$] to calculate the standard deviation of the "y" data points.

While you're doing any statistical analysis calculation, the data is collected in the memories as follows:

	0	**n** (The number of data points)
memory	**1**	Σy
	2	Σy^2
	3	Σx
	4	Σx^2
	5	Σxy
	7	Last x value entered, *plus one.*

Note also that the formula used for the variance is:

$$\text{Variance} = \frac{\Sigma(x_i - \overline{x})^2}{N} \text{ or } \frac{\Sigma(y_i - \overline{y})^2}{N}$$

The symbol Σ just means "the sum of".

If you make a mistake entering any data point, you can remove the bad points by re-entering them as before, but pressing [INV] before [2nd] [Σ+] as you key it in.

Example: You're grading an exam, and the scores are in. You'd like to see how well the class has done. The scores are tabulated below:

96	65	81
85	76	86
57	98	75
78	100	72
81	70	80

Press	**Display/Comments**
[INV] [2nd] [C.t]	**0** Be sure to clear the entire machine
96 [2nd] [Σ+]	**1.** The calculator counts
85 [2nd] [Σ+]	**2.** your data points for you
continue for all points	.
	.
	.
72 [2nd] [Σ+]	**14.**
80 [2nd] [Σ+]	**15.**
[2nd] [x̄]	**80.** Class average
[2nd] [σ2] [√x]	**11.564313** Standard deviation.

SOME SPECIAL NOTES ON STATISTICAL FUNCTIONS

Occasionally, when you're dealing with the analysis of 2 sets of related data, you'll need to "step up" the x variable by one, for each "y" variable you enter. (For example, for data collected by year — 1961, 1962, 1963, etc.). In this case you only need to enter the first x value, and your calculator will automatically increment x by one for each y value entered. This feature is useful in programming what are called "trend line analyses" — which folks into statistics know about. Also, the variance calculations above all use "N weighting" — which as folks who use statistics know, is suitable for analyzing *population* data. Statisticians use (N-1) weighting when analyzing *sample* data. To obtain (N-1) weighting in variance calculations, you can use the following key sequence after calculating the variance from the keyboard:

$$\boxed{\times}\ \boxed{\text{RCL}}\ 0\ \boxed{\div}\ \boxed{(}\ \boxed{\text{RCL}}\ 0\ \boxed{-}\ 1\ \boxed{=}\ .$$

This multiplies by $\frac{n}{n-1}$
(Note that the square root of the variance is the standard deviation.)

THE TRACKMAKERS: A TOUR OF THE PROGRAMMING KEYS

Your calculator is a powerful problem-solving device, equipped with a variety of features that are ready to use – right from the keyboard. In addition, though, your calculator is programmable. This means you can *teach* your calculator to "push its own buttons" to automatically perform a variety of calculations (including those that involve decisions). You can teach your calculator up to 50 problem-solving steps, and then have it execute these steps as often as you like – with the touch of a single key! In this chapter we'll go on a tour of the keys on your machine especially devoted to programs and programming. If you already have some programming experience, you may be able to just take this brief tour, pick up your calculator and program it right away.

WHAT'S HAPPENING INSIDE:

The way your calculator "learns" a program is in reality quite simple. There's a special *memory*, called a "program memory" inside your machine, that remembers the program keystrokes you teach it. As you program your machine, each keystroke sequence you enter is stored in order in this memory as a simple code. When you've finished entering a program, your calculator can then go back, read the "codes" in order, and push its own buttons for you in the exact sequence you've taught it.

To enter a program, you use the programming keys clustered on the left side of your keyboard. These keys let you teach the calculator the necessary problem-solving keystrokes. When you've taught your calculator a program, it's as if you've set up "tracks" for it to run on – a path for it to follow in solving your problem. When the calculator is programmed and ready to go, you can start it off at the beginning of these tracks and, much like a train, run down the line of steps you've entered. The "track" is your program – stored in program memory and the "train" is called the *program pointer*. As the train on the program track comes to (or points to) each step you've programmed, the calculator pushes its own buttons for you as it was directed.

This chapter, then, will give you a quick review of the "trackmakers" – or programming keys of your calculator.

LRN – THE LEARN KEY

Pressing this key once puts your calculator in the "learn" mode of operation – ready to be taught a program. When the calculator is in "learn" mode, the display takes on a whole new format, displaying 4 (and sometimes 5) digits.

00 00

The left two digits in the display will indicate the *step number* you are about to enter into the machine. (You can enter up to 50 steps, labelled 00 through 49.) The digits to the right are for key codes. These are two and three-digit numbers that tell the calculator which program steps you're entering. These right two digits (and a 3rd one that will pop up as you enter certain keystrokes) will be zero *while you enter a program*. This is because the display is always reading out the *step you are ready to enter*, and immediately moves to the next step location in program memory after you enter a step. (Later we'll be discussing how you can go back and check the key codes for each step you've entered.)

Pressing the LRN *key a second time takes your calculator out of "learn" mode*, and it's ready to run a program – or do calculations right from the keyboard. At this point the display changes back to its standard format.

Note that while you're in "learn" mode, the calculator won't do keyboard calculations – it will interpret each keystroke you enter as a program step to be stored. Once you've stored a program and get out of "learn" mode, you can do any keyboard calculations you need to – without affecting the program stored in program memory. Also, note that turning your calculator OFF clears the program memory completely (as well as all other calculator memories and registers).

RST – THE RESET KEY

This is the "start back at the beginning" key. It tells the program pointer to go back to step 00. This key can be used as part of a program (you can enter it while in "learn" mode), or right from the keyboard when out of "learn" mode. When encountered as part of a program, the program pointer immediately jumps back to the very first step – step 00, and keeps on going. When used from the keyboard when *out* of "learn" mode, pressing RST just sets the program pointer to step 00.

The RST also has other special functions. It resets the subroutine level counter back to zero and clears the subroutine return register.

R/S – The Run/Stop Key

Pressing the R/S key (when *out* of "learn" mode) tells your calculator to *start* running a program (from the current location of the program pointer). If a program *is* running, pressing this key (and holding it momentarily) tells it to stop. The R/S key has a dual function – if things are stopped, it starts them going; if a program is running, it stops it.

The R/S key can be used as part of a program, too – when entered *in "learn" mode.* If a program is running and it encounters an R/S instruction as a program step, it simply stops and displays whatever is currently in the display register. This allows you to stop a program to see intermediate results, to enter new numbers the program may need to work on, or to halt the action when the program is finished.

Note that when you're out of "learn" mode and press R/S, the program pointer begins stepping through program memory *starting from its current location.* If you want to start at the first program step (step 00), you need to press RST before pressing R/S. (You can position the program pointer at any other location you'd like to start at with the GTO key – discussed later.)

2nd Pause – The Pause Key Sequence

This key sequence is designed to be used as part of programs, and you enter it into your program while in "learn" mode. While a program is running and the program pointer comes to a 2nd Pause instruction, it stops the program for about ¾ of a second and displays whatever result is in the display register at that point. Then, it continues on to the next step.

The 2nd Pause key sequence lets you get a quick look at a program result, without stopping the action. This sequence is especially handy when used in repetitive programs (or "loops") – letting you watch the changing results of continuing calculations. You can watch your money "grow" in a compound interest account as the months go by, or watch a function go to a "limit", etc. If you'd like a pause that's *longer* than ¾ of a second, you can just add additional 2nd Pause key sequences, and keep things in the display ¾ of a second longer for each pause sequence you add. (Keep in mind, however, that each 2nd Pause key sequence you add to a program will use up one of your 50 allowed program steps.)

2nd Lbl n – The Label Key Sequence

This key allows you to *label* any point in a program (or an entire program), or a subroutine (subroutines will be discussed later). When in "learn" mode, you press the 2nd Lbl **n** key sequence, where **n** is any digit **0** through **9**. This lets you label up to 10 points in a program. (Label numbers cannot be repeated in the same programs.)

Labels in a program are convenient, because they work with other keys (GTO and SBR), to let you place the program pointer at any point in a program you select. Labels don't interfere with any program steps or calculations in progress – they are just reference "signposts" for you and the program pointer. You should put a label at any key point in your program – especially if you are going to want the program pointer to go (or "transfer") to that point as your program runs. If you label a program point, you can start your program running from that point easily by using the GTO key (we'll discuss this in a moment).

Merged Codes:

The entire 2nd Lbl **n** key sequence takes only one program step, because the calculator stores both the instruction and label digit with a special shortened code called a *merged* code. Try this on your calculator: Turn it OFF and then ON, press LRN, then 2nd Lbl. The display now reads:

00 86 0

The first two digits are the step number: **00**, the next two digits are the code for "label": **86**, and the last digit (zero) is the location for the label number (**0-9**). You would now enter a label digit to complete the label – say 3. When you press 3 the display reads: **01 00**, which is the next step location for program entry.

There are several program key sequences that use this merged code format. Basically, these are any of the functions that require a label number or memory location to be complete. (All the memory instructions STO RCL, etc. are examples.) Whenever you see the third rightmost digit light up while you're programming in "learn" mode, the calculator is requesting a label or memory number (0-9). (Don't forget to put one in – and, also, keep a record of what each label or memory is being used for.) Note that the 2nd Fix **n** key sequence also uses a merged code.

GTO n – The "Go To" Key Sequence

This key sequence can be used as part of a program (when entered *in* "learn" mode), or right off the keyboard (when *out* of "learn" mode). Pressing GTO **n**, where **n** is a *label number* (**0** through **9**) immediately moves the program pointer to the label number **n**. In a program, when the program pointer encounters a GTO **n** instruction, it goes right to label **n**, and continues on from that point. (The GTO **n** sequence is stored as one program step – with a "merged" code.)

The GTO key creates what is called an "unconditional transfer" when you use it in a program. No matter what's going on, when the program pointer comes to a GTO **n**, it goes right to the program label **n** and continues running. There are other types of program transfers made possible by combining the GTO **n** key sequence with other keys. These other keys let you place a *condition* right before the GTO **n** instruction. When you use these other keys , you can build decision points into your program – and create what are called *conditional transfers.*

The GTO **n** sequence is handy in a variety of situations. You can, for example, write a long program that performs several different functions. If you *label* each function with its own label number, you can easily use each function individually – without running the entire program. When *out* of "learn" mode just press GTO **n** (the label number of the function you want), and then start the program with the R/S key. The program pointer will start up from the labeled point.

GTO 2nd nn – The "Go To a Step Number" Key Sequence

This key sequence can be used only when *out* of "learn" mode. It simply lets you position the program pointer at any step number you choose. When out of "learn" mode, if you press GTO 2nd , and then any two digit program step number (00-49), the program pointer immediately moves to that step number. This feature is especially handy if you want to change or correct a program step (see editing keys – discussed later); or if you'd like to start the program pointer from some unlabeled point in a program.

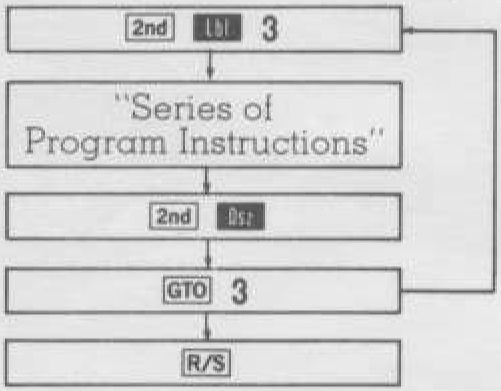
Try this on your calculator. Turn it OFF, then ON, and press GTO 2nd **16**. Now press LRN . The display reads **16 00**, telling you that you've moved the program pointer to step location 16. In similar fashion you could locate the program pointer at any step: 0 through 49. (Trying to go beyond step 49 results in a flashing error indication).

2nd Dsz – THE "DECREMENT AND SKIP ON ZERO" KEY SEQUENCE

This key sequence is a powerful one. It lets you set up a repetitive calculation (or "loop") for as many times as you select. (You store the number of repetitions in memory zero). What this sequence really does is allow you to control whether or not the program pointer moves on to the step that immediately follows it, or *skips* that step and goes on to the next one. When the program pointer comes to a 2nd Dsz key sequence in a program here's what happens:

- First, the contents of memory zero are decreased by one. (This assumes that a positive integer is in memory 0. If a *fractional* number is in memory zero, the calculator "acts" as if the *next* larger number is in memory zero. If a *negative* number is in memory zero, it is incremented by one.)
- The calculator then asks: is the content of memory zero *equal* to zero?
- If the answer is *no*, the program pointer proceeds right on to the step following 2nd Dsz .
- If the answer is *yes*, the program pointer *skips the following* instruction and continues.

The step that immediately follows the 2nd Dsz key sequence can be anything, but a GTO **n** instruction is often handy at this point for setting up repetitive calculations. For example, in the program situation shown in the diagram, the "series of program instructions" would be executed repetitively because of the loop.



The number of *times* the "instructions" were executed would be controlled by the number stored in memory zero (stored before running the program). If a 6 were stored in memory zero, the series of instructions would be carried out 6 times, then the program would halt.

Note that the series of instructions would also be executed 6 times if a 5.2, a −6, or a −5.2 were stored in memory zero before the program pointer arrived at this series of instructions. Also, be aware of the fact that since memory zero is involved whenever 2nd Dsz is used, it cannot be used for other storage purposes while handling a 2nd Dsz sequence, although the number in memory 0 can be used as part of the computational sequence by recalling it.

INV 2nd Dsz – THE "DECREMENT AND SKIP IF NOT ZERO" KEY SEQUENCE

This key sequence works in the same fashion as the previous one, except the condition of the skip is reversed. When the program pointer comes to an INV 2nd Dsz key sequence, here's what happens:

- One is subtracted from the contents of memory zero. (Again, it's assumed that the number in memory zero is a positive integer. If it's fractional, it acts as if the *next* largest integer is in memory zero. If the number in memory zero is negative, one is *added* to it.)
- The calculator then asks, is the content of memory *zero equal to* zero?
- If the answer is *no*, the program pointer *skips* the instruction immediately following the INV 2nd Dsz sequence.
- If the answer is *yes*, the program pointer proceeds right on to the instruction immediately following the INV 2nd Dsz .

This instruction is also handy for setting up repetitive calculations in a variety of programing situations.

CONDITIONAL TRANSFERS

There is a family of instructions on your calculator designed to let you build decision-making into your programs. The basic operation of this family of instructions is similar – and they're quite easy to learn, but they are also flexible enough to allow a great deal of programming power.

Basically when a *conditional transfer* key sequence is encountered in a program, the calculator immediately compares two quantities: the number in the display register (called x) and the number in a special test or "t" register (called t). (The "t" or test register is just memory 7 on your calculator.) The calculator then asks one of 4 questions about how x and t compare – depending on the program instruction you use:

2nd x=t	Is x = t? (Is x equal t?)
INV 2nd x=t	Is x ≠ t? (Is x not equal t?)
2nd x≥t	Is x ≥ t? (Is x greater than or equal to t?)
INV 2nd x≥t	Is x < t? (Is x less than t?)

If the answer is yes, the program pointer goes on to the instruction immediately following. If the answer is NO, the program pointer SKIPS the instruction immediately following and continues.

Let's go through each of the conditional transfer instructions (and the keys for using them) one at a time.

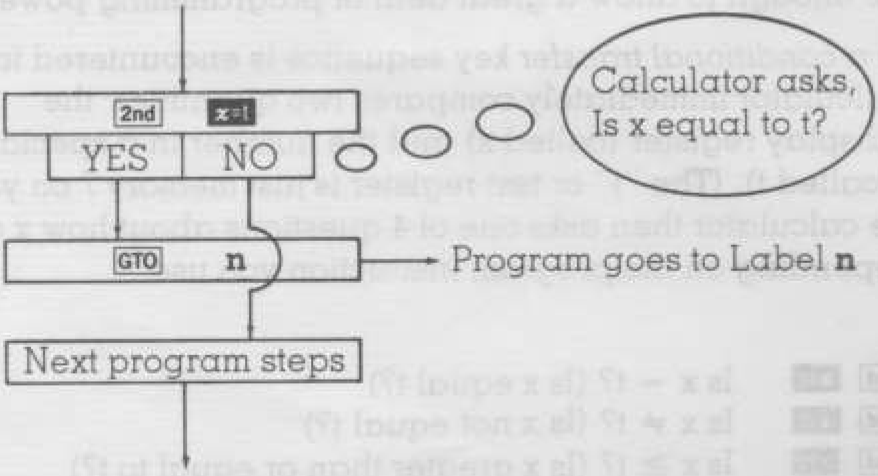
x⇄t – The "x exchange with t" Key

When this key is pressed the contents of the display register and the "t" register (memory 7) are just exchanged. This provides a convenient method of getting any test, or t, value into the test register. (You can also use the 2nd Exc 7 key sequence, or simply the STO 7 sequence if it is appropriate in your program.)

2nd x=t – The "Is x Equal to t?" Key Sequence

You can enter this key sequence while in "learn" mode to have the program pointer take one of several pathways in your program. When the program pointer comes to this instruction in a program, the calculator compares what's in the display register (x) and what's in the "t" register (t) and asks: Is x equal to t? If the answer is *yes*, the calculator goes right on to the step following the 2nd x=t key sequence. If *no*, the pointer *skips* whatever instruction follows and moves on to the next instruction and continues.

The instruction that immediately follows the 2nd x=t key sequence can be anything, but you'll find a GTO **n** key sequence especially useful, as shown in the diagram below:



Again, key things to remember when using the 2nd x=t key sequence:

- The calculator asks the question: Is x = t?
- If the answer is *yes* — it goes on to the step immediately following.
- If the answer is *no* — it *SKIPS* the step following and goes on.

OTHER CONDITIONAL TRANSFERS:

INV 2nd x=t – "Is x Not Equal to t?" (x ≠ t) Key Sequence

This key sequence is entered into a program (while in "learn" mode) to create a conditional program transfer (or decision). When the program pointer encounters the "Is x *not* equal to t?" key sequence — INV 2nd x=t — the following things happen:

- The calculator compares the contents of the display register (x) to the contents of the test register (register 7).
- The calculator asks: Is x *not* equal (or unequal) to t?
- If the answer is *yes*, the program pointer *goes right on* to the step that immediately follows.
- If the answer is *no*, the program pointer *SKIPS* the following step and continues.

Again note that the program step immediately following the INV 2nd x=t key sequence can be any program step you choose, though a GTO **n** is especially handy at these points.

2nd x≥t – "Is x Greater Than or Equal to t?" Key Sequence

When the program pointer comes to this instruction in a program, the following things happen:

- The calculator compares the contents of the x and t registers.
- It asks the question: "Is x greater than or equal to t?"
- If the answer is *yes,* it goes on to the instruction that immediately follows.
- If the answer is *no,* it *skips* the following instruction and continues.

INV 2nd x≥t – "Is x Less Than t?" – (x<t) Key Sequence

When the program pointer comes to this instruction in a program, the following things happen:

- The calculator compares the contents of the x and t registers.
- It asks the question: "Is x less than t?"
- If the answer is *yes,* it goes *on* to the step immediately following the INV 2nd x≥t key sequence.
- If the answer is *no,* it *skips* the step immediately following, and continues.

Additional Notes:

It's really quite easy to remember these instructions if you keep your eye on the "big picture". Basically you can have the calculator make any one of four tests: Is x equal t, is x greater than or equal t, or their inverses – Is x unequal t, Is x less than t? Whatever the case, the calculator makes a test and asks a question. If the answer to the question is *yes,* it goes right on to the step immediately following. If the answer is *no,* it *skips* the step immediately following, and continues.

Note that the quantity x that's compared to t is the *contents of the display register.* The display register holds numbers to 11 digits, of which only 8 are actually shown (correctly rounded) in the calculator's display window. In certain cases these last 3 digits can affect the outcome of the comparisons described in this section. (These cases are rare, but if a problem should arise, it's helpful to know about this possibility.)

SUBROUTINES

Using subroutines is not only a handy technique for saving program steps and breaking a large program down into easy to analyze segments: it's also just plain good programming practice. A subroutine is just a "miniprogram" you can write as part of any program. Anytime you need to use a set of program steps more than once in a program, it's a good idea to make that series of steps into a subroutine. Here's how:

To make any series of program steps into a subroutine, all you need to do is:

- Start the series of program steps with a *Label.*
- End the series of steps with an INV SBR key sequence.

Subroutines can be of any length, as long as your entire program with all its subroutines is no more than 50 steps.

To Use a Subroutine in a Program:

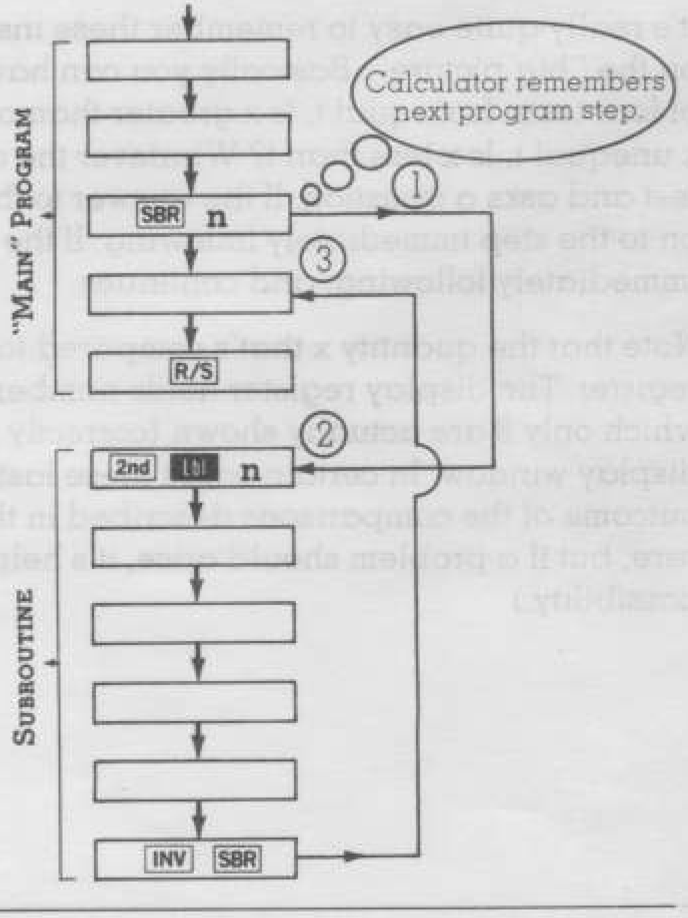
- While in "learn" mode, anytime you need to use any subroutine, just press: SBR **n,** where n is the label of the subroutine. You can use (or "call", as is sometimes said), a subroutine as many times as you'd like in a program.

When the program pointer comes to a SBR **n** key sequence in a program, the calculator does the following things for you (illustrated in the diagram):

① First, it "notes" the step in the program where the SBR **n** instruction occurs in a special memory (called the subroutine return memory).

② The program pointer goes right to label **n** and starts executing the instructions after it, until it comes to the INV SBR instruction.

③ When the program pointer reaches the INV SBR instruction, it returns to the point where the subroutine was "called", and continues.



To Use Subroutines Right from the Keyboard:

One advantage of using subroutines in a program is that you can use any subroutine in the program separately from the program itself. To do this, just press SBR **n** while *out* of "learn" mode. The program pointer will immediately go to label **n** and execute the subroutine. In this case the INV SBR key sequence at the end of the subroutine will act just like an R/S instruction.

Notes on Subroutines:

Using subroutines is *good programming practice!* Subroutines let you organize your programs easily (and thus can help you find errors – if they exist.) It's also easier for other programmers to understand and use your programs when they're in bite-size subroutine "chunks".

Your calculator is equipped with *two levels* of subroutine capability. This means that while you're inside one subroutine, you can "call" or use another subroutine. You can't go to a third level, however. If a second level subroutine tries to "call" a third, a flashing error condition will be created. The "subroutine return" memory has room for only two return locations.

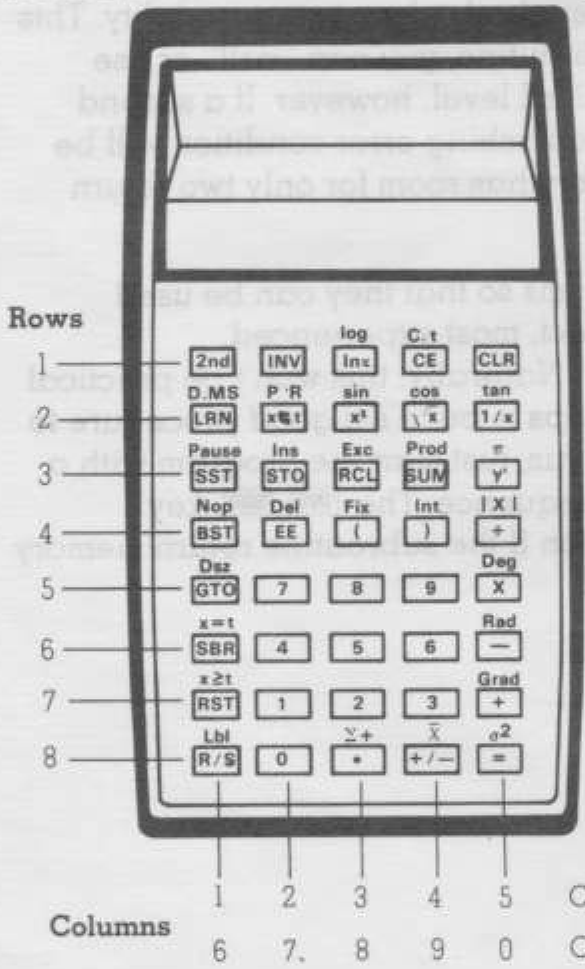
You can easily write any of your programs so that they can be used as subroutines in other programs. (In fact, most experienced programmers suggest that you do this.) Naturally, this won't be practical for programs that contain close to 50 steps – but it's a good procedure to follow for any shorter programs. To do this, just start the program with a label, and end it with an INV SBR key sequence. The INV SBR key sequence acts just like an R/S instruction if the subroutine return memory is empty.

PROGRAM EDITING: KEY CODES

Your calculator is equipped with a series of keys that allow you to alter your programs easily and rapidly to fit new problem solving situations, or to correct any errors you may have in your program. These *editing* features of the calculator let you get to any point in a program, analyze it, change a step, add or delete steps, or leave "blanks" in the program for later use.

As was mentioned earlier in this chapter, your calculator stores the steps in any program as a sequence of key codes. These codes are arrived at in a pretty straightforward fashion. The code for any *number* key on the keyboard is just the number itself (00 through 09). The key code for any first function key is a two digit number: The first digit is the row the key is in, the second digit is its column. (Rows are numbered from 1 to 8 top to bottom, columns from 1 to 5 left to right, as shown in the diagram.)

For example, the key code for the STO instruction is 32, x^2 is 23. For second functions, just add 5 to the column number — with the rightmost column being indicated by a zero. The code for 2nd sin is 28, the code for 2nd π is 30.



When you first enter a program in "learn" mode, you won't see the key codes, because the program memory display always moves ahead to the next step. To see the codes in any program, you need to enter the program first, then go back and examine the codes. Here's how to do this:

Let's enter a simple program to work with. We'll enter a program that displays the *squares* of the counting numbers (1, 2, 3, ... etc.) Key in the program as follows:

Press	**Display/Comments**
OFF-ON	0 Clears entire machine
LRN	00 00
1	01 00
SUM	01 34 0
2	02 00
RCL	02 33 0
2	03 00
x^2	04 00
2nd Pause	05 00
RST	06 00
LRN RST	0

To run this program, just press R/S and you should see 1, 4, 9, 16, etc. Press R/S and hold it momentarily to stop the program. Now, using this program you can check on the key codes and edit, using the keys and features described on this and the following page. (Press RST now to get back to the beginning of the program.)

SST – Single Step Key

When used in "learn" mode, this key steps through your program one step at a time, and lets you check codes. Note that this key has *no* keycode of its own, and can't be entered as a program step. This key can also be used when *out* of "learn" mode, to *execute* a program one step at a time. For editing you press SST repeatedly, to see the codes for your program:

Press	**Display/Comments**	
LRN	00 01	1
SST	01 34 2	SUM 2
SST	02 33 2	RCL 2
SST	03 23	x^2
SST	04 36	2nd Pause
SST	05 71	Last program step RST
SST	06 00	

PROGRAM EDITING:
BST
GTO 2nd nn
2nd Nop 2nd Ins

BST – The Backstep Key

This key operates in "learn" mode only, and lets you step *back* through your program one step at a time. Press BST repeatedly to go back through your program codes. (This key also has no code of its own and can't be entered as a program step.)

GTO 2nd nn

As was already mentioned, this key sequence can be used when **out** of "learn" mode to go directly to any program step. Press LRN to get out of "learn" mode, then press GTO 2nd **05.** Then press LRN again. The display shows **05 71** – you're at step 5 in the program which is the RST instruction.

To Change or Edit a Program:

You can write over program steps, blank out program steps, insert steps or delete steps. Let's take a quick look at each of these:

To Write Over a Program Step:

Just get to the exact step you want to write over, and simply enter the new keystroke or keystroke sequence. The new instruction will write over the old one and replace it. (The display will move on to the next step as soon as you do this, so you can go back and check the new code with the BST key.)

2nd Nop – The "No Operation or NO-OP" Key Sequence

This sequence lets you blank out any instruction in your program with a "NO-OP" instruction that's simply ignored by the calculator. The 2nd Nop key sequence lets you leave blank spaces in a program you can fill in later, or lets you "blank out" any instruction you wish. Just step to the instruction you want to blank out while in "learn" mode, and press 2nd Nop. A NO-OP instruction (code 46) will replace the old one.

2nd Ins – "Insert" Key Sequence

This sequence allows you to insert additional steps in a program easily. If you want to add instructions to your program, step to the program step that's right at the point you'd like to add additional steps (while in "learn" mode).

Then press 2nd Ins.

This moves all instructions from that point on down by one location, and leaves a blank space at the location you're at. Now you can add a new instruction. To add more instructions, just repeat the process. *Note:* Any step at location 49 will be "pushed off" and lost when you press 2nd Ins.

2nd Del – "DELETE" KEY SEQUENCE

This sequence allows you to remove an instruction from a program, and close up the gap – moving all the following instructions up by one location. Just step to the exact instruction you want to delete, and press the 2nd Del key sequence.

STEPS IN WRITING A PROGRAM:

As you learn more about programming, you'll begin falling into a "natural rhythm" where programming solutions will begin to suggest themselves to you as soon as you're confronted with a problem. The actual steps you'll follow in getting to a program solution will naturally depend on your personal approach, but these steps suggest one way to proceed:

1. Study the problem – gather the equations and procedures you'll need.
2. Set a destination for yourself. Determine how you'll use the program, so that when the program is complete, you'll know what to enter, what keys you'll press, and what you want to see displayed.
3. *Plan the Program "Route"* – Conceptualize how the program pointer will "flow" through the program (a flow chart or schematic diagram of the steps you'll follow may be helpful here).
4. Write down the actual program steps (or, as you gain confidence, key them right into your machine). As you go along keep careful track of what is stored in each memory, and what any label you use signifies. These are items that are easy to forget, and forgetting them is a common source of error and "start overs".
5. Once the program is entered, check it with known test data to be sure it's working.
6. Edit and correct as necessary.
7. Document or "write up" the program carefully to save it for future use. Standard program record forms are provided for this purpose and the form is arranged to follow the "steps in writing a program" that we've discussed here. Once your program is working, you can press RST, LRN and then use the SST key to step through the key codes and record them on the program record form. This type of complete record will let you (or a friend) pick up the form at a future date and understand the program – how it works, how to use it, *and* how to check it to be certain it's working.

DEVELOPMENT OF PROGRAMMING STYLE:

Whether you're programming your calculator, or a large scale computer – keep one point in mind: there is no single unique programming solution to any problem. As you gain experience in programing, you'll find yourself developing your own unique style, "tricks of the trade", and favorite techniques in getting at a solution. As you begin gaining experience, it's a good idea to review your techniques periodically to be sure you're using all the power of your machine. Don't be afraid to explore new routes, try new alternatives, and experiment with new methods. Programming is an excellent exercise in clear, common sense, logical thinking that many folks enjoy (and some are quite addicted to). Your calculator is deliberately designed to allow you to get started easily, and then grow quite quickly into an expert in creating programs.

MOVING ON:

One more point – your calculator is easy to use, and readily accessible and affordable. *But*, it incorporates *most all of the major features and functions of any large scale computer!* The concepts of a program memory, a program pointer that moves through it, labels, go to instructions, loops and conditional transfers are common to all digital programmable devices. Once you've learned to program your calculator you'll find it quite easy to move on to programming any machine – no matter what the scale. Larger scale computers simply accept their input in different ways, have more ways to get output results to you, and in general are equipped to handle and store larger amounts of information. Most commonly a beginner will approach these machines by writing instructions one at a time, on cards, paper tape or magnetic discs. These one-at-a-time instructions will resemble your keystroke instructions – particularly in most common beginning computer languages such as FORTRAN or BASIC. The keyboard design of your calculator will allow you to learn programming easily in either of these languages, after learning the basics on your machine.

In the next decade, logic and memory will become omnipresent in our everyday lives. Programmable devices will be everywhere – in the kitchen, in your home finances, in communications, in personal security, – everywhere. Learning the basics of programming on your calculator *now*, will prepare you for the sound use of these devices – and enable you to enjoy them as a *natural* part of your daily living.

BASIC TROUBLESHOOTING

Even the most careful person makes mistakes from time to time. Your calculator is easy and straightforward to use, but sometimes through a programming oversight — even when you've carefully keyed things in — your program won't work properly. Here are a few tips and things to look for in troubleshooting basic programs.

Your calculator will let you know about errors in a program in one of several ways. If at any point in a program an inaccurate key sequence or series of operations creates an error condition (see *Appendix* C), *the program will stop* and show you a flashing display. This simply means that you've asked the calculator to do something in a program that it couldn't normally do even from the keyboard.

Exceeding the display limits or using an incorrect key sequence will "stop the music". In other cases you may *not* see a flashing error condition, but the results of your program are just plain wrong. It's always good practice to verify the accuracy of a program with test input data, whose program outcome you *know*. If you don't get the result expected, it's time to look for a problem.

When a program is not working — try to "think like the machine". Go through your flow chart (and key codes) and execute — *blindly* — whatever it says to do there. You may find that you've inadvertently boxed your program into some corner. The SST key can be of help — both in and out of "learn" mode, in tracking just what may have gone wrong.

RST — Be sure you've reset the calculator wherever you need to — to get back to the beginning of the program. Also, be sure to turn the machine OFF and ON before beginning a new program — to clear everything.

— Algebraic Operating System

Remember that all the power of the **AOS™** algebraic entry system is working for you — even inside programs. All operations will be carried out following the correct mathematical hierarchy. This order *may not* correspond to the order of entry you've used in your programs.

Remember that your calculator will hold up to *4 pending operations*, at any one time.

If you have any doubts about the order in which expressions are being evaluated — remember that you can insert parentheses — ( ) — to be certain that things are evaluated in the order you specify.

Remember that special (single variable) function keys must *follow* the number they work on. If you want the sine of 30° at some point in a program, the keystroke sequence is 30 [2nd] [sin] .

[=] Key – this completes any and all pending operations waiting to happen inside your machine – be careful using it. (Be careful of using [=] in subroutines – discussed later. It's better to use [(] [)] keys to complete calculations when in a subroutine, because [=] will complete calculations in the main program as well.)

Labels – Be sure you use any one of the 10 labels *no more than once* in any program.

Angular mode – If you're using any of the trig functions, remember that when you first turn the calculator on, it interprets all angles as degrees, unless you change things with the [2nd] [Rad] or [2nd] [Grad] key sequences, and stays there until you change again (or turn the calculator OFF-ON).

Memories – Be careful to check and see that all operations using memories are working with the *correct memory*. Also, note that if you have 4 operations pending in any expression in your program – that memories 5 and 6 will be used to hold some of the expressions. Any numbers stored previously in memories 5 and 6 will be lost if a long expression involving 4 pending operations is entered later on. (Also, memory 7 is the "t" register – you'll be learning more about it in the next section.) Memory zero is the memory used with the [2nd] [Dsz] key sequence – keep that in mind, too.

You'll find that with a little experience your programs will run accurately and with little trouble. You'll also develop an "eagle eye" for a potential problem area once it's hit you one time – so the best way to learn troubleshooting is to keep using your calculator. After awhile you probably will rarely need to troubleshoot any of your programs.

APPLICATIONS

- ON SALE.
- COUNT OFF.
- FACTORIAL.
- COMBINATION.
- EVALUATING INTEGRALS.
- LINEAR REGRESSION.
- HI - LO

ON SALE!

In this case let's say you've got a job in a department store. The manager thinks that running sales is a good idea. In fact, he has his employees marking down price tags throughout the store quite often. He also likes to vary the amount of the discount from department to department, 25% for shoes, 15% in hardware, etc. In this programming situation we'll review some of what you've seen in the first example and expand on using memories as part of programs.

DESTINATION

You'd like to be able to go to any department in the store that you're assigned, key in the price of any item on the shelf, press R/S and get the discounted price. You'd also like to be able to vary the amount of the discount as easily as your manager changes the discount amount from department to department.

PLANNING THE ROUTE

When an item is discounted by 25%, you can find the new price using the formula: New price = old price × (1-.25). In fact, for any discount amount you can find the new price from this same formula:

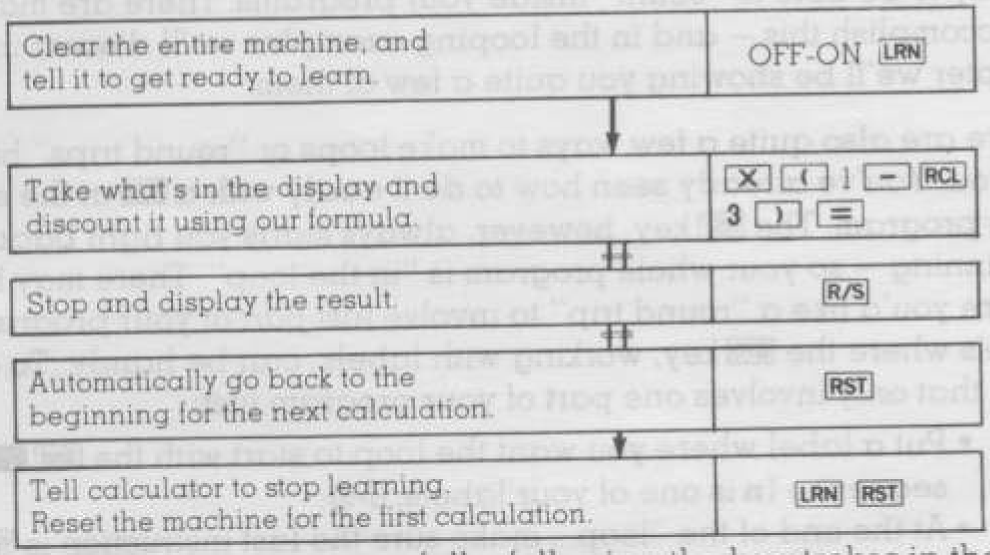
New price = old price × (1 – discount)

In our program, we'll store the discount amount as a decimal number (25% = 0.25) in memory 3. Then we'll recall it as we need to in calculating the new price.

New price = old price × (1- RCL **3**).

The advantage of doing this in your program is that if you go to a new department with a new discount rate, all you'll need to do is store the new rate in memory 3, then proceed. This technique is a handy one in many situations where you'd like to change a number in a program without rewriting the program itself.

MAKING TRACKS



Clear the entire machine, and tell it to get ready to learn.	OFF-ON LRN
Take what's in the display and discount it using our formula.	× (1 − RCL 3) =
Stop and display the result.	R/S
Automatically go back to the beginning for the next calculation.	RST
Tell calculator to stop learning. Reset the machine for the first calculation.	LRN RST

Enter your program carefully, following the keystrokes in the diagram.

RUNNING IT

The manager has decided to take 25% off shoes selling for $14.95, $17.99 and $18.99. Then he'd like you to mark 13% off books selling for $6.00, $7.99 and $12.50. What should the new prices be?

PRESS	DISPLAY/COMMENTS
2nd fix 2	**0.00** This tells the machine to round the display value to the nearest cent
0.25 STO 3	**0.25** Store the first discount rate as a decimal in memory 3
Enter old prices & press R/S to see new ones:	
14.95 R/S	**11.21**
17.99 R/S	**13.49** New prices
18.99 R/S	**14.24**
0.13 STO 3	**0.13** Store the new discount rate
Enter prices discounted for 13%, & press R/S to see new prices:	
6.00 R/S	**5.22**
7.99 R/S	**6.95** New prices
17.50 R/S	**15.23**

Note in this case that by storing the discount rate in memory 3, it's very easy to change rates as you move from department to department.

NEXT STOP

You'd like to keep track of all the marking up and down you do all day. Could you rewrite the above program to keep track of the *number* of items you've marked down in memory 4?

COUNT OFF! LOOPS WITH GTO n AND 2nd lbl n

You've already seen – and will continue to see – how it's often handy to be able to "count" inside your programs. There are many ways to accomplish this – and in the looping examples we'll discuss in this chapter we'll be showing you quite a few of them.

There are also quite a few ways to make loops or "round trips" happen for you. You've already seen how to do it easily with a RST at the end of your program. The RST key, however, always starts you right back at the beginning – so your whole program is "in the loop". There may be times where you'd like a "round trip" to involve just *part* of your program. Here's where the GTO key, working with labels, can be handy. To create a loop that only involves one part of your program just:

- Put a label where you want the loop to start with the 2nd lbl **n** key sequence (**n** is one of your labels, 0-9).
- At the end of the "loop", make sure the last instruction is GTO **n** (where n is the same label number).

Here's a counting example that involves loops:

DESTINATION

You'd like to cook up a program that sets your calculator counting by 1's, 2's, 4's, etc. – whatever number you'd like. You'd also like to *start* counting *from* any number you select. The way you'd like things to work for you is as follows:

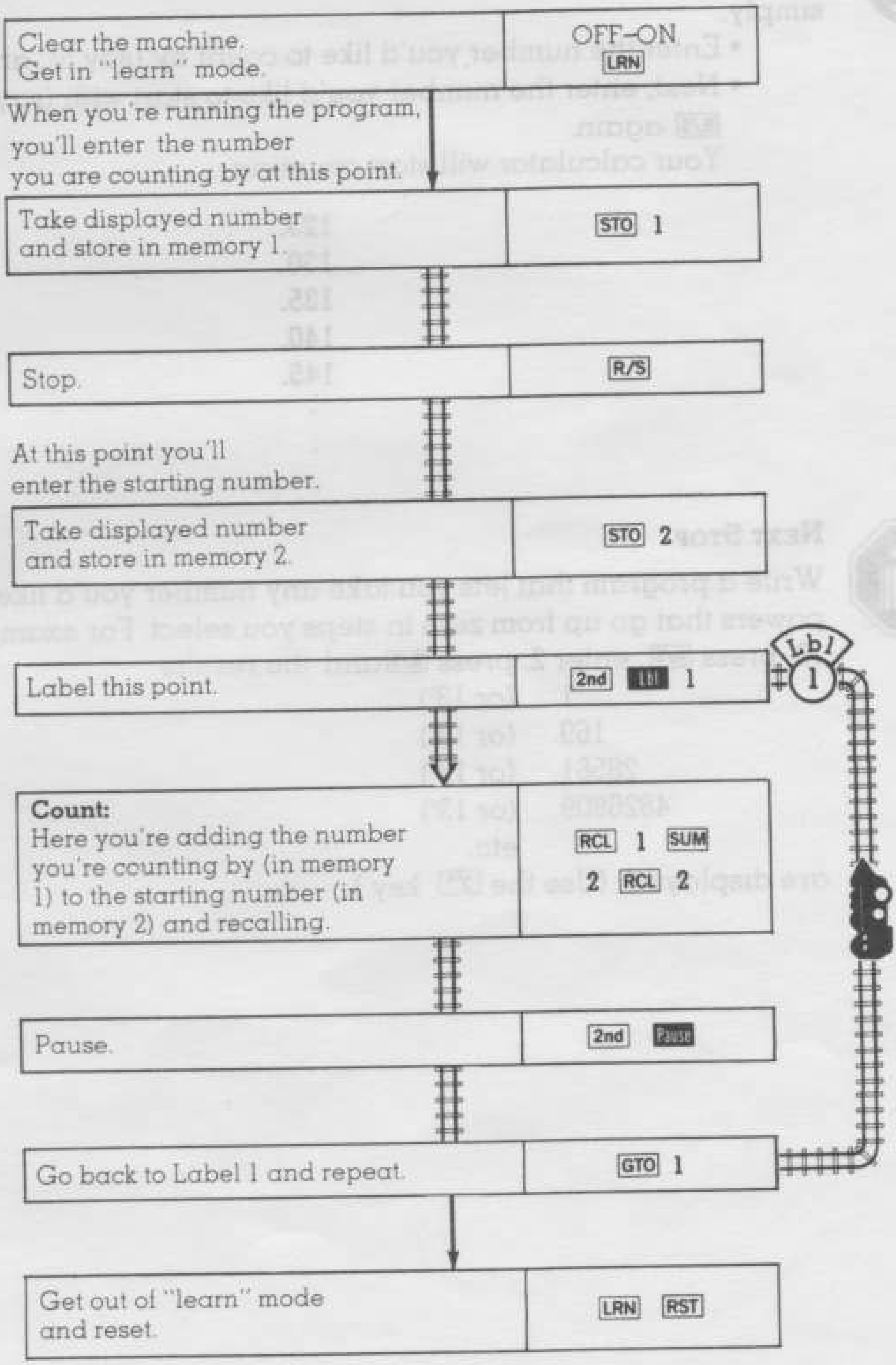
- You'll enter the number you'd like to *count by*, and press R/S
- You'll enter the "starting number" and press R/S . . .
  The calculator will then count by whatever number you've selected, beginning at your "starting number".

PLANNING THE ROUTE

In the first part of the program we'll store the first number entered – the number we're counting by – in memory 1. Then we'll store the *starting* number for our counting in memory 2. Now, after you've stored these two numbers, you won't want to return to this portion of the program for the counting sequence, so put down a label – telling the calculator this is the point to come back to in the "loop".

After the label, write the "counting" portion of the program. One way to do this is to recall the number you're counting by (in memory 1), and sum this to the starting number (in memory 2). This is the first number in the counting sequence, so recall memory 2 and pause to look at it. Then just repeat the process by returning to the label.

MAKING TRACKS



Clear the machine. Get in "learn" mode.	OFF–ON LRN

When you're running the program, you'll enter the number you are counting by at this point.

Take displayed number and store in memory 1.	STO 1
Stop.	R/S

At this point you'll enter the starting number.

Take displayed number and store in memory 2.	STO 2
Label this point.	2nd Lbl 1
Count: Here you're adding the number you're counting by (in memory 1) to the starting number (in memory 2) and recalling.	RCL 1 SUM 2 RCL 2
Pause.	2nd Pause
Go back to Label 1 and repeat.	GTO 1
Get out of "learn" mode and reset.	LRN RST

Running It

Enter the program carefully from the flow chart. To start counting, simply:

- Enter the number you'd like to *count by* (say 5), and press R/S.
- Next, enter the number you'd like to start with (say 120), and press R/S again.

Your calculator will start counting:

125.
130.
135.
140.
145.
.
.
.

Next Stop

Write a program that lets you take any number you'd like, and raise it to powers that go up from zero in steps you select. For example, you enter a 13, press R/S, enter 2, press R/S and the results

1. (or 13^0)
169. (or 13^2)
28561. (or 13^4)
4826809. (or 13^6)
etc.

are displayed. (Use the y^x key.)

FACTORIAL!
LOOPS WITH 2nd Dsz GTO n

An idea that crops up often in studies of probability and statistics is what's called the *factorial* of a number. The *factorial* of a number is indicated with an exclamation point "!". So, "17 factorial" is written 17!. If you've ever wondered how many ways 17 people could arrange themselves in 17 chairs – the answer if 17!. The factorial of any number, say 5, is calculated by multiplying the number "out" as follows:

$5! = 5 \times 4 \times 3 \times 2 \times 1$

The factorial of any number, N, is equal to:

$N! = N \times (N-1) \times (N-2) \ldots\ldots \times 1.$

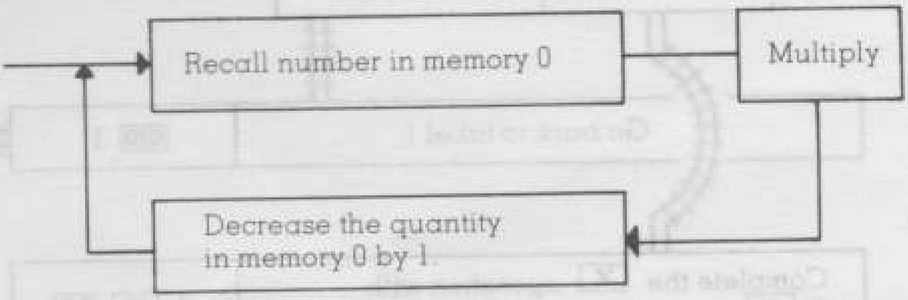
(The number 0! is defined to be 1.)

DESTINATION

Write a program that allows you to enter a number (say 17), press R/S, and have its factorial calculated and displayed.

PLANNING THE ROUTE

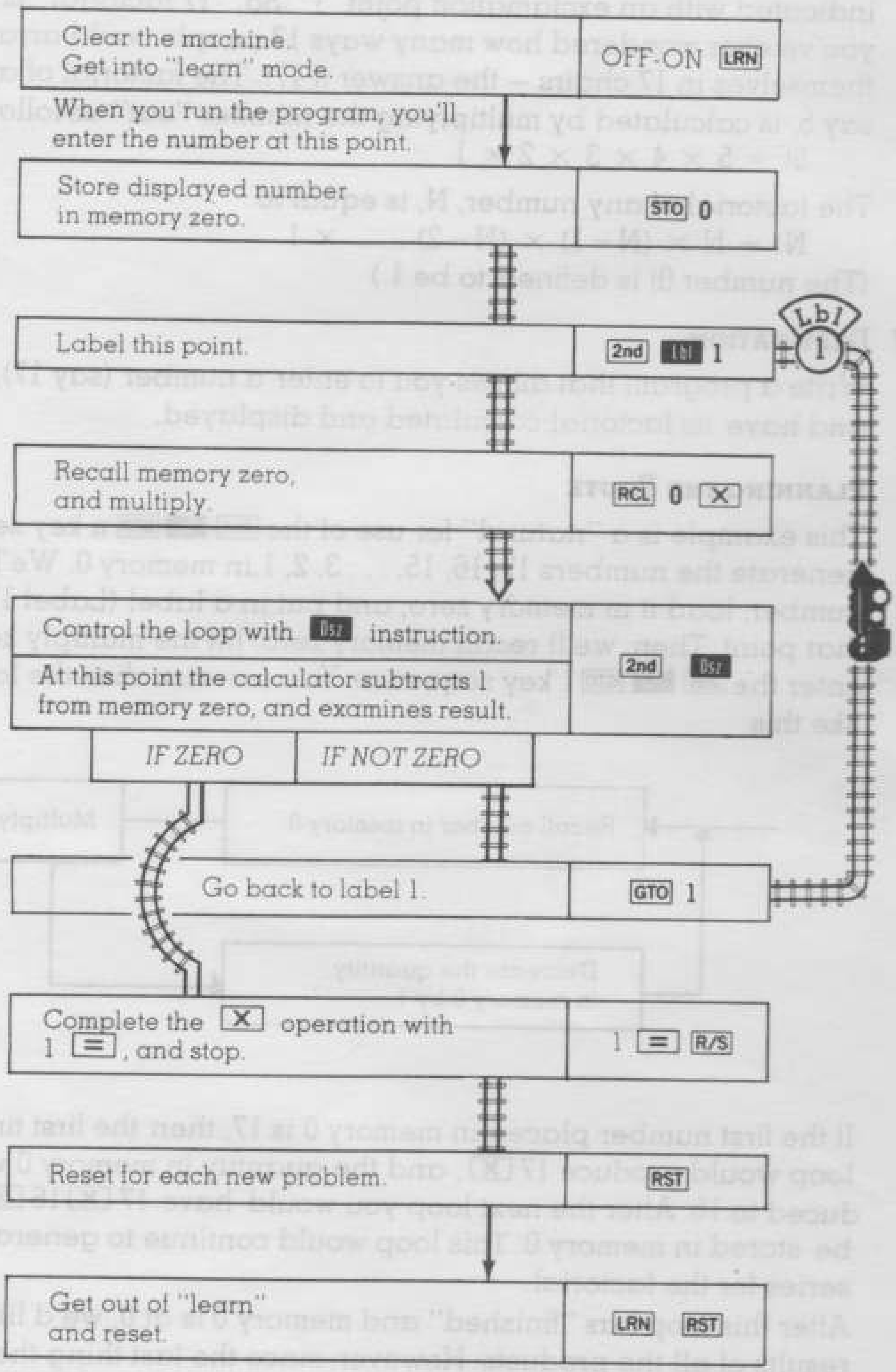
This example is a "natural" for use of the 2nd Dsz GTO **n** key sequence to generate the numbers 17, 16, 15, . . . 3, 2, 1, in memory 0. We'll enter our number, load it in memory zero, and put in a label (Label 1) right after that point. Then, we'll recall memory zero, hit the multiply key and then enter the 2nd Dsz GTO **1** key sequence. You can visualize the loop something like this:


Recall number in memory 0
Multiply
Decrease the quantity in memory 0 by 1

If the first number placed in memory 0 is 17, then the first time around the loop would produce 17 ×, and the quantity in memory 0 would be reduced to 16. After the next loop you would have 17 × 16 ×, and 15 would be stored in memory 0. This loop would continue to generate the required series for the factorial.

After this loop has "finished" and memory 0 is at 0, we'd like to see the results of all the products. However, since the last thing the calculator is told to do is ×, an = cannot immediately follow as × = produces an error signal. To avoid this, you can put a "1 =" at the end of the program, and the "final product" × 1 = gives the correct result.

MAKING TRACKS: FACTORIAL PROGRAM


Clear the machine.
Get into "learn" mode.
OFF-ON LRN
When you run the program, you'll enter the number at this point.
Store displayed number in memory zero.
STO 0
Lbl
1
Label this point.
2nd Lbl 1
Recall memory zero, and multiply.
RCL 0 ×
Control the loop with Dsz instruction.
At this point the calculator subtracts 1 from memory zero, and examines result.
2nd Dsz
IF ZERO
IF NOT ZERO
Go back to label 1.
GTO 1
Complete the × operation with 1 =, and stop.
1 = R/S
Reset for each new problem.
RST
Get out of "learn" and reset.
LRN RST

Running It

Enter the program carefully from the flow chart. Once entered, to calculate the factorial of any number (positive integer less than 70), just enter it and press R/S.

Some examples:

Press	**Display/Comments**
5 R/S	**120.** or 5!
10 R/S	**3628800** or 10!
17 R/S	**3.5568743 14** or 17!
50 R/S	**3.0414093 64** or 50!

Notice that this last calculation keeps your calculator busy for quite some time. Remember any time you put your machine into a loop – give it time to crank out the result. Some loops can take hours – even days! Note also that this program will give you an incorrect result for 0! If you enter 0 and press R/S – the result is zero, not 1 as it should be. You might want to develop a program where you do not get an incorrect result for 0!.

As we've already mentioned, using the 2nd Dsz key sequence will come in handy in a variety of situations. In this section we've introduced a few of its most straight-forward uses. In later sections we'll be speaking more on the "ins and outs" of its application.

Next Stop

Develop a program that displays the factorial of the numbers 1, 2, 3, 4, . . . etc. You'd like each integer displayed for two pauses, then its factorial displayed for three.

COMBINATIONS:
SBR AND INV SBR

Let's consider a program example from statistics that involves the use of subroutines. A "combination" calculation arises when things are being put together in groups (say 10 items grouped 3 at a time without repeating). The general formula for the number of ways n things may be combined, r at a time, is written $\binom{n}{r}$ or ${}_nC_r$ and equals: $\frac{n!}{(n-r)!r!}$

(remember that n! means "n factorial" – see *Chapter 4*).

Destination

You want a program that allows you to enter the total number of the things you are looking at (n) and press R/S ; then enter the number you want in each group r, press R/S ; and the calculator will display the number of combinations of n things taken r at a time (${}_nC_r$).

Planning the Route

To compute the factorial, we'll use a subroutine that works with the 2nd Dsz key sequence as we described previously. We can "lift" the factorial program described in that example and use it as a subroutine here, by just starting it with a label, (label 2), and ending it with an INV SBR key sequence.

We'll calculate n!, store it in memory 1; calculate r!, store it in memory 2; calculate (n – r)!, store it in memory 3; then use these results to compute

$${}_nC_r = \frac{n!}{(n-r)!\,r!}$$

Making Tracks

Enter the program carefully from the flow chart on the next page.

Running It

Using your program,calculate the number of poker hands (5 cards) that can be dealt from deck of 52 cards (${}_{52}C_5$).

Press	Display/Comments
52 R/S	**8.0658175 67** (This is 52!) *Note:* Allow time for your calculator to finish.
5 R/S	**2598960.** the number of possible poker hands from a 52 card deck (${}_{52}C_5$).

Now let's say you would like to use *just* the subroutine part of the program to find 10!

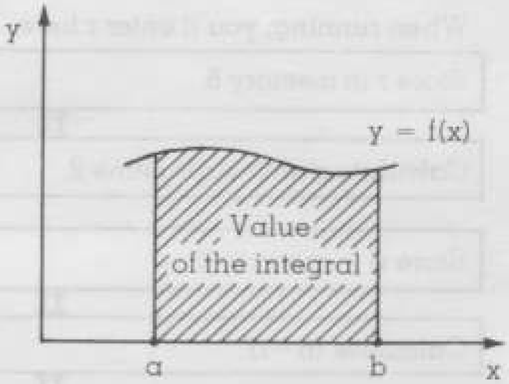
10 SBR **2** **3628800.** (this is 10!)

Making Tracks

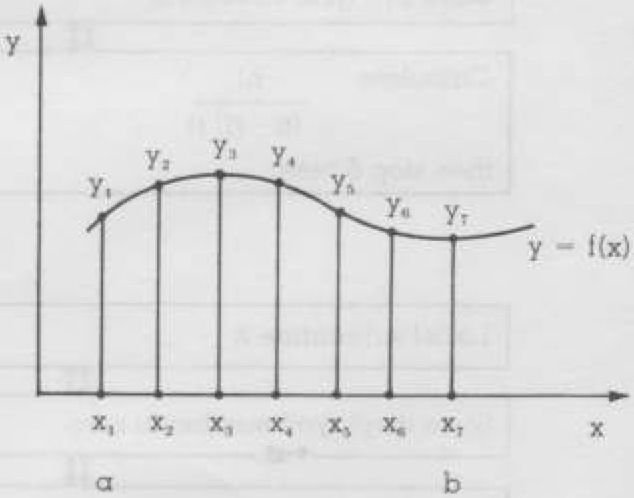
Clear the entire machine. Get into "learn" mode.	OFF-ON LRN	00
When running, you'll enter n here.		
Store n in memory 5.	STO 5	01
Calculate n! with subroutine 2.	SBR 2	02
Store n!, and stop.	STO 1 R/S	04
When running, you'll enter r here.		
Store r in memory 6.	STO 6	05
Calculate r! with subroutine 2.	SBR 2	06
Store r! in memory 2.	STO 2	07
Calculate (n − r).	RCL 5 − RCL 6 =	11
Calculate (n − r)! with subroutine 2.	SBR 2	12
Store (n − r)! in memory 3.	STO 3	13
Calculate $\frac{n!}{(n-r)!\,r!}$ then stop & reset.	RCL 1 ÷ (RCL 3 × RCL 2) = R/S RST	23
Label subroutine 2.	2nd Lbl 2	24
Store displayed number in zero.	STO 0	15
Calculate factorial. (see program in *Chapter 4*).	2nd Lbl 1 RCL 0 × 2nd Dsz GTO 1 1 =	32
End of subroutine.	INV SBR	33
Get out of "learn" and reset.	LRN RST	

EVALUATING INTEGRALS: SIMPSON'S RULE

If you've had any calculus courses, you're probably familiar with the integral symbol "$\int$". This symbol stands for "the integral of", and $_a\int^b f(x)dx$ is read "the integral of f(x) from a to b" – and represents the *area* under the curve $y = f(x)$ from point a to point b as shown in the figure.



If you know the techniques of integral calculus, you know that there are methods allowing you to evaluate this integral exactly. You also know that in some cases this evaluation may be extremely difficult, or even impossible! In these cases, approximation techniques are sometimes used. One of these approximation methods is called *Simpson's approximation.*



When using Simpson's approximation (or Simpson's rule, as it's often called), you divide the area under the curve into an *even* number of parts (n). The width of each part, w, is then given by: $w = \frac{b-a}{n}$. In the figure shown, we've divided the area into 6 parts, so $w = \frac{b-a}{6}$. To compute the approximate total area under the curve (the approximate value of the integral), you use the following formula:

$$\text{Area} = \frac{w}{3} \times (Y_1 + 4Y_2 + 2Y_3 + 4Y_4 + \ldots . + 2Y_{n-1} + 4Y_n + Y_{n+1}).$$

This formidable looking formula is actually quite easy to evaluate with the program we'll develop in this section.

Here Y_1 = the Y value with $x = a$
Y_2 = the Y value with $x = a + w$
Y_3 = the Y value with $x = a + w + w$,
⋮
etc.
⋮
Y_{n+1} = the Y value with $x = b$

As an example, we'll write a program for (and evaluate) the integral of $y = 2x^2 + 3x - 1$ from $a = 1$ to $b = 5$, using $n = 6$. (Note, the integral approximation gets more accurate as n is made larger.)

DESTINATION

We'll construct a program with a subroutine that repeatedly solves the equation $y = 2x^2 + 3x - 1$, for each value of x between a and b. Then using these values Simpson's approximation will be computed. The program will be constructed so that you'll:

- Enter n (number of divisions – must be an even integer). and press STO **0**.
- Enter b and press STO **1**.
- Enter a and press STO **2**.
- Press RST R/S, and after a wait, the display will read the integral of the function. To evaluate the integral of any new function all that must be altered will be subroutine 1, which evaluates the function. Just press GTO **1** while out of LEARN mode, press LRN, enter the new function (remembering that x is in memory 2) and end with an INV SBR. Press LRN again, enter your new values for a, b, and n and continue.

PLANNING THE ROUTE

Since $y = f(x) = 2x^2 + 3x - 1$ needs to be evaluated many times in this program, and needs to be easily changeable so that other integrals can be evaluated, we'll place it in a subroutine at the end of the program. We'll begin the program by calculating $w = \frac{b-a}{n}$, storing it and recalling as we need it. We'll include program steps to find the right multiplying number (called coefficients) for each "Y" term in the Simpson's rule formula:

$$\text{Area} = \frac{W}{3}(Y_1 + 4Y_2 + 2Y_3 + 4Y_4 + \ldots + 2Y_{n-1} + 4Y_n + Y_{n+1}).$$

(Note that the coefficients of Y will either be 1, 2, or 4 depending on which term we're evaluating.) We'll use subroutine 1 to evaluate each Y term, multiply by the correct coefficient and sum the result to memory 3. Then we'll step to the next x value, determine a new coefficient, a new Y value, and repeat for n steps. At the end of these steps, we'll compute Y_{n+1}, sum it to 3, multiply by $\frac{W}{3}$ and display the final result.

MAKING TRACKS

Enter the program carefully from the flow diagram on the following page.

RUNNING IT

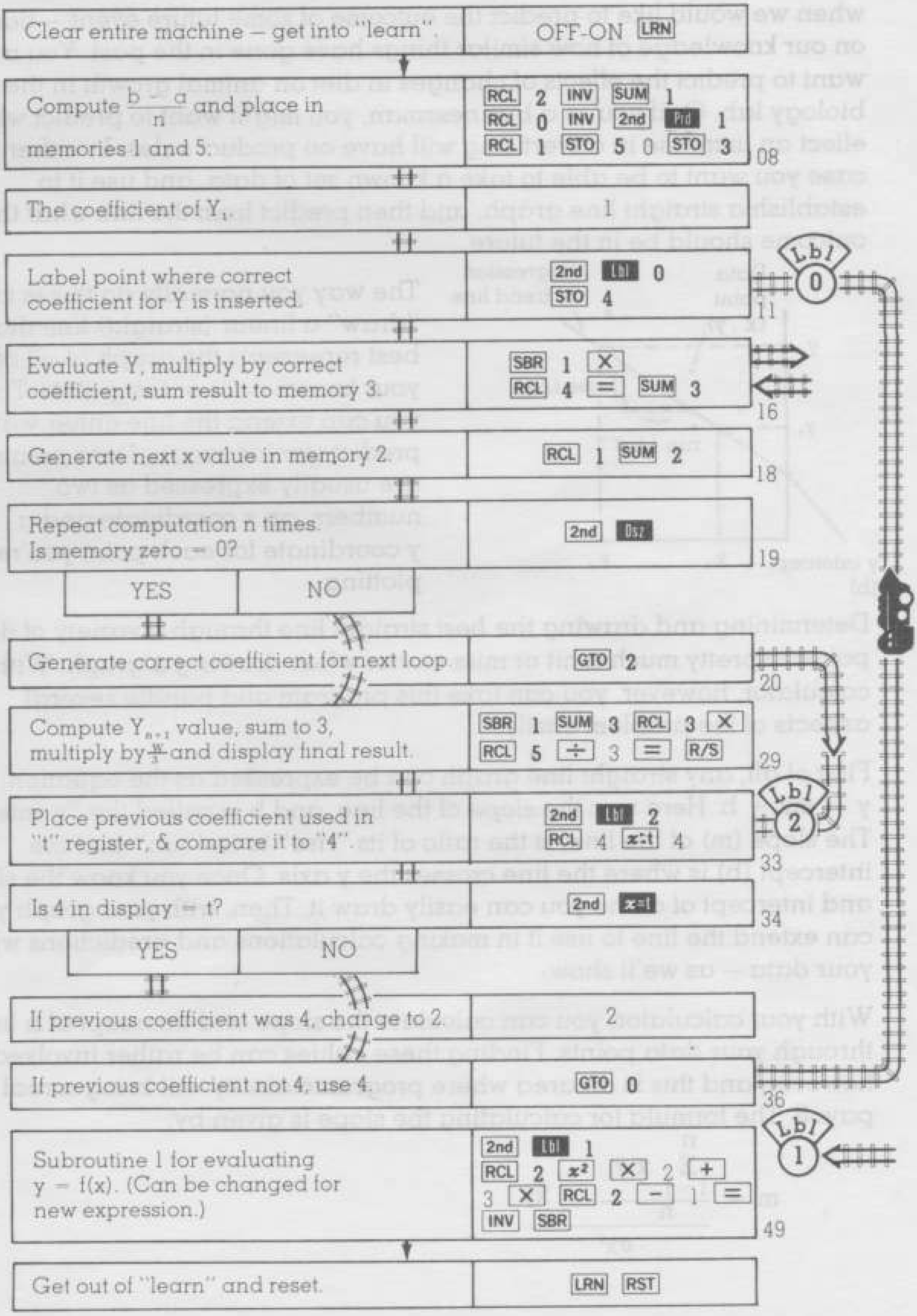
Again, keep in mind that this program evaluates the integral of $y = 2x^2 + 3x - 1$. To change to another function:

- Be sure you're out of "learn".
- Press GTO **1**.
- Press LRN.
- Enter the new function.
- End with an INV SBR sequence.

To evaluate $2x^2 + 3x - 1$ for $a = 1$, $b = 5$, $n = 6$:

PRESS	**DISPLAY/COMMENTS**
6 STO **0**	**6.** Enter n,
5 STO **1**	**5.** Enter b,
1 STO **2**	**1.** Enter a,
RST R/S	**114.66667** The approximate integral value — give your calculator some time to finish.

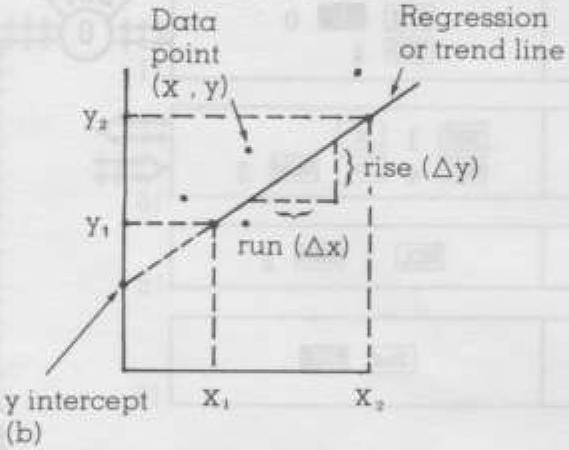
Making Tracks



Clear entire machine – get into "learn".	OFF-ON LRN	
Compute $\frac{b-a}{n}$ and place in memories 1 and 5.	RCL 2 INV SUM 1 RCL 0 INV 2nd Prd 1 RCL 1 STO 5 0 STO 3	08
The coefficient of Y_1	1	
Label point where correct coefficient for Y is inserted.	2nd Lbl 0 STO 4	11
Evaluate Y, multiply by correct coefficient, sum result to memory 3.	SBR 1 × RCL 4 = SUM 3	16
Generate next x value in memory 2.	RCL 1 SUM 2	18
Repeat computation n times. Is memory zero = 0?	2nd Dsz	19
YES	NO	
Generate correct coefficient for next loop	GTO 2	20
Compute Y_{n+1} value, sum to 3, multiply by $\frac{\Delta x}{3}$ and display final result.	SBR 1 SUM 3 RCL 3 × RCL 5 ÷ 3 = R/S	29
Place previous coefficient used in "t" register, & compare it to "4".	2nd Lbl 2 RCL 4 x⇌t 4	33
Is 4 in display = t?	2nd x=t	34
YES	NO	
If previous coefficient was 4, change to 2.	2	
If previous coefficient not 4, use 4.	GTO 0	36
Subroutine 1 for evaluating y = f(x). (Can be changed for new expression.)	2nd Lbl 1 RCL 2 x² × 2 + 3 × RCL 2 − 1 = INV SBR	49
Get out of "learn" and reset.	LRN RST	

LINEAR REGRESSION

As most of us progress through school or business, there will be times when we would like to predict the outcome of some future event – based on our knowledge of how similar things have gone in the past. You might want to predict the effects of changes in diet on animal growth in the biology lab. Or if you're a businessman, you might want to predict what effect an increase in advertising will have on product sales. In either case you want to be able to take a known set of data, and use it to establish a straight line graph, and then predict from the line what the outcome should be in the future.



The way you normally do this is to "draw" a linear (straight) line that best represents the graph of all of your known data or test points. Then you can extend the line either way to predict similar events. Data points are usually expressed as two numbers, an x coordinate and a y coordinate for each point you're plotting.

Determining and drawing the *best* straight line through a variety of data points is pretty much a hit or miss matter when drawing a graph. With your calculator, however, you can take this program and handle several aspects of the problem easily.

First of all, any straight line graph can be expressed as the equation: $y = mx + b$. Here m is the *slope* of the line, and b is called the "y intercept". The slope (m) of the line, is the ratio of its "rise" to its "run" and the intercept (b) is where the line crosses the y axis. Once you know the slope and intercept of a line you can easily draw it. Then, with your graph you can extend the line to use it in making calculations and predictions with your data – as we'll show.

With your calculator, you can calculate the slope and intercept of a line through your data points. Finding these values can be rather involved, however, and this is an area where programmability will bring a real payoff. The formula for calculating the slope is given by:

$$m = \frac{\dfrac{\sum_{i=1}^{n} x_i y_i}{n} - \overline{x}\overline{y}}{\sigma x^2}$$

The formula for the intercept is:

$b = \bar{y} - m\bar{x}$

$\sum_{i=1}^{n} x_i y_i$ means the sum of all data coordinates multiplied together, starting at 1 and going to n; (x_1y_1, x_2y_2, ...). With the help of some special keys on your calculator and the program we'll develop, you'll be able to solve for m and b with a fraction of the effort normally involved in handling these complex formulas.

The program will also allow you to use your data to make predictions. Once you find m and b, you'll be able to enter a new "x" value and get its corresponding "predicted" "y" value on the line using the formula $y = mx + b$. Or you can enter a "y" value and "predict" the corresponding "x" value on the line using the equation $x = \frac{y - b}{m}$.

This program also lets you find a number called the coefficient of determination (labeled r^2). The number r (computed from r^2 with a simple $\boxed{\sqrt{x}}$ keystroke) is called the absolute value of the correlation coefficient. This is a measure of how well your data points correspond or correlate to your line. This helps determine the accuracy of predictions you make from your line. A value close to 1 indicates a high correlation. (The calculator could fit your points accurately to a trend line.) A value close to zero indicates that the sets of data have a low correlation and the trend line developed by the calculator is not as reliable for predictions. The formula for r^2 is:

$$r^2 = \frac{m^2 \sigma_x^2}{\sigma_y^2}.$$

The program we're about to embark on involves many, many calculations and will pretty well "load up" your calculator. The important point is that with your calculator handling the difficult mathematics, you can use the powerful tools of linear regression for prediction and planning, without getting stopped by the mass of computation involved.

DESTINATION

We'll develop a program that allows you to enter all of your data points easily, and then compute m, b, and r^2. Then you'll want to be able to compute a "y" value for any given "x" value, or for any given y to find a corresponding "x" value. To do all of this you'll follow these steps:

- Enter data points by entering x, pressing x:t, then entering y and pressing R/S. You'll do this until all of your (x, y) data points are entered.
- Then to find m, press SBR **0**.
- To find b, press SBR **1**.
- To find r^2, press SBR **2**. Then $\sqrt{x}$ to find r.
  (Remember a "r" value near one means good correlation, a value near zero means that the calculator could not reliably correlate your data to a straight line.)
- To predict what value on the line that y will have for a given "x" value, just enter the "x" value and press SBR **3**.
- To predict what value on the line that x will have for a particular y, enter the "y" value and press SBR **4**.

PLANNING THE ROUTE

The first step will be to enter all the data points using the special data collection keys on your calculator. These are the 2nd $\Sigma+$, 2nd $\bar{x}$, and 2nd σ^2 keys on the lower right of the machine, described at the end of *Chapter 2.*

Rather than pushing 2nd $\Sigma+$ for each entry, we'll include a simple sequence in the program to help in entering data points. Then we'll divide the rest of the program into five subroutines to calculate m, b, r^2, and to find x or y. Follow the flow chart carefully as steps have been conserved using memory arithmetic.

MAKING TRACKS

Key in the flow chart carefully from the *Making Tracks* chart on the next page.

MAKING TRACKS

Clears all and into "learn".	OFF-ON LRN		
(You'll enter x press x:t, then y & R/S)			
Enters each data point & resets.	2nd Σ+ R/S RST	03	
Identifies subroutine 0.	2nd Lbl 0		Lbl 0
Solve for the slope, m.	RCL 5 ÷ RCL 0 − 2nd $\bar{x}$ × INV 2nd $\bar{x}$ = ÷ INV 2nd $\sigma2$ = INV SBR	16	
Identifies subroutine 1.	2nd Lbl 1		Lbl 1
Calculates the y intercept, b.	SBR 0 × INV 2nd $\bar{x}$ +/− + 2nd $\bar{x}$ = INV SBR	25	
Identifies subroutine 2.	2nd Lbl 2		Lbl 2
Solves for r^2.	SBR 0 x^2 ÷ 2nd $\sigma2$ × INV 2nd $\sigma2$ = INV SBR	34	
You'll enter an x value.			
Subroutine 3.	2nd Lbl 3		Lbl 3
Finds y to correspond to entered value of x.	STO 7 SBR 0 2nd Prd 7 SBR 1 SUM 7 RCL 7 INV SBR	42	
You'll enter a y value.			
Identifies subroutine 4.	2nd Lbl 4		Lbl 4
Finds x to correspond to entered value of y.	STO 7 SBR 1 INV SUM 7 SBR 0 INV 2nd Prd 7 RCL 7 INV SBR		

Note: You won't need to leave "learn" as all 50 steps are used and the calculator leaves "learn" automatically.

Running It

Test Scores vs. Performance: An Example

Let's say you're a businessman and your sales manager is spending a considerably sum on a test for prospective sales employees. You'd like to see if this test is actually telling you anything about how well the employee will function in the field. Does a higher test score mean superior sales performance? How strong a *correlation* is there between these two factors in your business? If there is a good correlation, you'd like to know the slope and the intercept of the straight line, so you can use these values for future predictions. Also, you'd like to know specifically what a test score of 9 predicts in sales performance; and what value test score would indicate a sales performance of 75.

Let's say you have samples of the test scores for 10 employees, along with records on sales performance expressed as the percentage of the time that each employee exceeded his or her weekly sales goals last year. The data is tabulated below:

Employee	*Employee Test Score (x)*	*Employee Sales Performance (y)*
Jerry	5	10
Ross	13	30
Joe	8	30
Ralph	10	40
Mary	15	60
Gary	20	50
Dean	4	20
Carole	16	60
Ted	18	50
Alecia	6	20

Follow the steps below in entering the data, and analyzing it to make a prediction:

Press	**Display/Comments**	
INV 2nd C.t RST	Clear memories and RST to get started. Then enter your data.*	
5 x:t 10 R/S	**1.**	*Note:* your calculator keeps count of each data point you enter.
13 x:t 30 R/S	**2.**	
8 x:t 30 R/S	**3.**	
10 x:t 40 R/S	**4.**	
15 x:t 60 R/S	**5.**	
20 x:t 50 R/S	**6.**	
4 x:t 20 R/S	**7.**	
16 x:t 60 R/S	**8.**	
18 x:t 50 R/S	**9.**	
6 x:t 20 R/S	**10.**	You've entered your 10 data points.
		To find the slope
SBR **0**	**2.6837607**	= m
		To find the "y" intercept
SBR **1**	**6.1367521**	
		To find the correlation factor
SBR **2**	**0.7497339**	= r^2
√x	**0.8658717**	= r There is a fairly good correlation — the r value is reasonably close to +1.
		What sales performance would a test score of 9 predict?
9 SBR **3**	**30.290598**	Prediction: about a 30% sales performance above goal.
		What test score would indicate a 75% above goal sales performance record?
75 SBR **4**	**25.659236**	An employee with a score of about 26 should achieve a 75% above goal sales rating.

**Note:* If you make a mistake entering any data point, you can remove the bad point by re-entering it following this sequence: re-enter "x" value, press x:t. Then re-enter y value and press INV 2nd Σ+. Then proceed with corréct data points as shown above.

HI-LO

How many guesses will it take you to discover a "secret" number between 0 and 1023? You'll get a clue after each guess (you're told whether your guess was "high" or "low"). This program makes your calculator a game that's fun to have along on trips in the car, while camping – anywhere.

To Play the Game:

The "Hi-Lo" program uses an initial entry number (less than one) to generate a "secret number" for you to guess. Each time you guess, a plus 1 is displayed if your guess is too large, a minus one is shown if your guess is too small, or the secret number is flashed if you guess correctly. After you key in the program, you'll follow these steps:

- Press ⊡, then enter the time of the day, the number of the month, and the day of the month. (This gives a convenient starting number less than 1.) Now press R/S. At this point, your calculator "knows" its secret number; and it displays a zero – telling you it's ready to play.
- Now try to guess the number by entering your guess and pressing SBR **4**. If your guess is "Hi" a 1 is displayed; if low, a −1 is shown. If you guess right, the secret number flashes.
- To guess again, enter your new number and press SBR **4**.
- After you've guessed correctly, press RCL **1** to see the number of guesses! To play again, press CLR to stop the flashing and SBR **1** – telling the calculator to create another secret number for you to guess. Then, just enter your number, press SBR **4** and "guess away".

Making Tracks

Key in the program using the *Hi-Lo* keystroke chart. (Don't forget OFF-ON and LRN.)

Hi-Lo

KEY	*LOC*	*CODE*
OFF-ON LRN		
STO 0	00	32 0
2nd Lbl 1	01	86 1
0	02	00
STO 1	03	32 1
2nd π	04	30
+	05	75
RCL 0	06	33 0
=	07	85
y^x	08	35
8	09	08
−	10	65
2nd Int	11	49
=	12	85
STO 0	13	32 0
×	14	55
1	15	01
0	16	00
2	17	02
3	18	03
+	19	75
1	20	01
=	21	85
2nd Int	22	49

KEY	*LOC*	*CODE*
STO 7	23	32 7
CLR	24	15
R/S	25	81
2nd Lbl 4	26	86 4
STO 2	27	32 2
1	28	01
SUM 1	29	34 1
RCL 2	30	33 2
INV 2nd x=t	31	−76
GTO 8	32	51 8
2nd x=t	33	66
GTO 7	34	51 7
1	35	01
=	36	85
R/S	37	81
2nd Lbl 8	38	86 8
1	39	01
+/−	40	84
=	41	85
R/S	42	81
2nd Lbl 7	43	86 7
RCL 7	44	33 7
GTO 3	45	51 3
LRN RST		

RUNNING IT

Here's how a typical game might go:

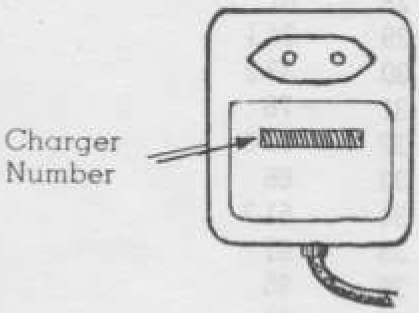
PRESS	DISPLAY/COMMENTS
	Enter random number (less than 1)of your choice.
.243411 RST	
R/S	**0** Secret number ready.
500 SBR **4**	**1.** Guess too high.
250 SBR **4**	**−1.** Guess too low.
400 SBR **4**	**−1.** Guess too low.
450 SBR **4**	**−1.** Too low.
475 SBR **4**	**1.** Too high.
470 SBR **4**	**1.** Too high.
466 SBR **4**	**466.** Flashing – you're right.
CLR RCL **1**	**7.** It took 7 tries.

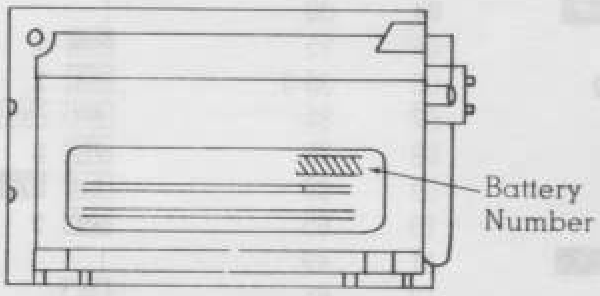
To use again, press CLR SBR **1**. Then enter your guess, press SBR **4** and you're going again.

APPENDIX A: BATTERY AND AC OPERATION

NORMAL OPERATION

Your calculator is designed for portable operation with periodic recharging of the battery pack with the adapter/charger supplied. It is important that the proper adapter/charger is used. If replacement of the battery pack or charger becomes necessary, be sure that an exact replacement is obtained.

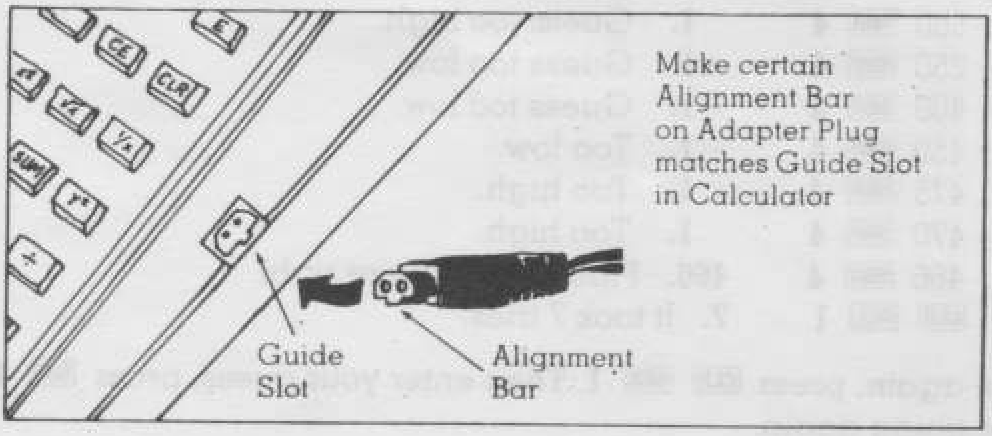




Note. Two different combinations of battery pack and adapter/charge may be used with your calculator. The BP/6 battery pack is used with the AC9900H-UK adapter/charger. The BP/7 battery pack is used with the AC9900R-UK adapter/charger.

Caution : Use of other than the proper Adapter/Charger may apply improper voltage to your calculator and damage the unit.

To ensure maximum portable operating time, connect the Adapter/Charger to a standard 240V/50 Hz outlet, plug into calculator, and charge battery pack at least 4 hours with the calculator OFF or 10 hours with the calculator ON. The adapter/charger and battery pack may become warm when used on AC power. This is normal and of no consequence. (**Caution** : The BP-6 battery pack should only be recharged while properly installed in your calculator.)


CE
CLR
SUM
Make certain Alignment Bar on Adapter Plug matches Guide Slot in Calculator
Guide
Slot
Alignment
Bar

When the battery pack is fully charged, the calculator will operate approximately 3 to 4 hours before recharging is necessary. However, don't hestitate to connect the adapter/charger if you know or suspect the battery pack is nearly discharged. A battery pack near discharge can adversely affect all calculator operations giving erroneous results. A discharged battery pack is typically indicated by a dim, erratic or blank display.

While individual cell life in a battery pack is difficult to predict, under normal use, rechargeable batteries have a life of 2 to 3 years or about 500 to 1000 recharge cycles.

PERIODIC RECHARGING

Although the calculator will operate indefinitely with the adapter/charger connected, the rechargeable battery pack can lose its storage capacity if it is not allowed to discharge occasionally. For maximum battery life, it is recommended that you operate the calculator as a portable at least twice a month, allowing the batteries to discharge, then recharge accordingly.

EXCESSIVE BATTERY DISCHARGING

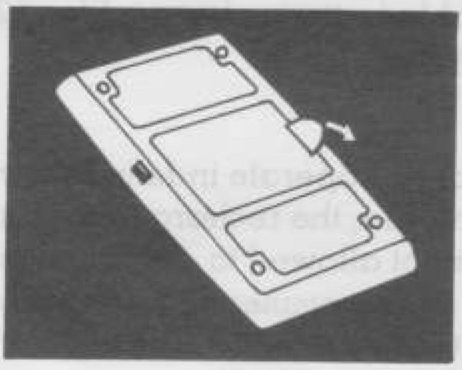
If the calculator is left on for an extended period of time after the battery pack is discharged (accidentally left on overnight, for example), connect the adapter/charger for at least 24 hours with the calculator OFF. If this does not restore normal battery operation the battery pack should be replaced. Repeated occurrences of excessive battery discharging will permanently damage the battery pack.

STORAGE

If the calculator is stored or unused for several weeks, the battery pack will probably need recharging before portable use. The battery pack will not leak corrosive material; therefore, it is safe to store the calculator with the battery pack installed.

BATTERY PACK REPLACEMENT

The battery pack can be quickly and simply removed from the calculator. Hold the calculator with the keys facing down. Place a small coin in the slot in the bottom of the calculator. A slight prying motion with the coin will pop the slotted end of the pack out of the calculator. Carefully disconnect the wires that attach the battery pack to the calculator. The pack can then be removed entirely from the calculator.



The metal contacts on the battery pack (where charger and calculator plug in) are the battery terminals. Care should always be taken to prevent any metal object from coming into contact with these terminals and shorting the batteries.

To re-insert the battery pack, first, attach the connecting wires to the terminals of the battery pack. Alignment should not be a problem as the connector will only fit in one position. Then, place the pack into the compartment so that the small step on the end of the pack fits under the edge of the calculator bottom. A small amount of pressure on the battery pack will snap it properly into position. (*Do not force.* It will fit easily when properly oriented.)

APPENDIX B: IN CASE OF DIFFICULTY

In the event that you have difficulty with your calculator, the following instructions will help you to analyze the problem. You may be able to correct your calculator problem without returning it to a service facility. If the suggested remedies are not successful, contact the Consumer Relations Department by mail or telephone (refer to "If You Have Questions or Need Assistance" later in this appendix). Please describe in detail the symptoms of your calculator.

Symptom	**Remedy**
1. Display is blank for no obvious reason.	Press and hold R/S momentarily. If display returns, the calculator was running a long program or operating in a continuous program loop. The battery pack may be discharged or improperly installed. Also, check to be sure the ON-OFF switch is fully in the ON position.
2. Display shows erroneous results, flashes erratic numbers, grows dim, or goes blank.	The battery pack is probably discharged or improperly connected. Refer to Battery and AC Operation in appendix A.
3. Display flashes while performing keyboard operations.	An invalid operation or key sequence has been pressed or the limits of the calculator have been violated. See Appendix C for a list of these conditions.

If none of the above procedures corrects the difficulty, return the calculator and charger PREPAID and INSURED to the applicable SERVICE FACILITY.

If the calculator is out of warranty, service rates in effect at time of return will be charged. Please include information on the difficulty experienced with the calculator as well as return address information including name, address, city, state and zip code. The shipment should be carefully packaged, adequately protected against shock and rough handling and sent to one of the Texas Instruments Service Facilities listed with the warranty.

For Technical Assistance

For technical questions such as programming, specific calculator applications, etc., you can call Bedford : 0234-63181. As an alternative, you can write to :
One of our addresses in Europe
(See back cover of this manual)

Because of the number of suggestions which come to Texas Instruments from many sources containing both new and old ideas, Texas Instruments will consider such suggestions only if they are freely given to Texas Instruments. It is the policy of Texas Instruments to refuse to receive any suggestions in confidence. Therefore, if you wish to share your suggestions with Texas Instruments, or if you wish us to review any calculator program key sequence which you have developed, please include the following statement in your letter :

> "All of the information forwarded herewith is presented to Texas Instruments on a nonconfidential, nonobligatory basis; no relationship, confidential or otherwise, expressed or implied, is established with Texas Instruments by this presentation. Texas Instruments may use, copyright, distribute, publish, reproduce, or dispose of the information in any way without compensation to me."

APPENDIX C: ERROR CONDITIONS

A flashing display indicates that the limits of the calculator have been violated or that an invalid calculator operation has been requested. Pressing $\boxed{\text{CE}}$ or $\boxed{\text{CLR}}$ stops the flashing. $\boxed{\text{CLR}}$ also clears the display and pending operations. $\boxed{\text{CE}}$ stops the flashing only, permitting further calculations with undisturbed pending operations. The display flashes for the following reasons:

1. Calculation entry or result (in display or memories) outside the range of the calculator, $\pm 1 \times 10^{-99}$ to $\pm 9.9999999 \times 10^{99}$. The exceeded limit is flashed, indicating underflow or overflow.
2. Inverse of a trigonometric function with an invalid value for the function such as $\sin^{-1}x$ with x greater than 1. The invalid value x is flashed.
3. Root or logarithm of a negative number. The root or logarithm of the absolute value of the number is flashed to indicate the sign error.
4. Raising a negative number to any power (or root). The power (or root) of the absolute value of the number is flashed.
5. Pressing two operation keys in succession. This affects $\boxed{+}$, $\boxed{-}$, $\boxed{\times}$, $\boxed{\div}$, $\boxed{y^x}$, and $\boxed{\text{INV}}$ $\boxed{y^x}$ (for $\sqrt[x]{y}$). The last entered number is flashed.
6. Pressing $\boxed{=}$ or $\boxed{)}$ after $\boxed{+}$, $\boxed{-}$, $\boxed{\times}$, $\boxed{\div}$, $\boxed{y^x}$, or $\boxed{\text{INV}}$ $\boxed{y^x}$ (for $\sqrt[x]{y}$). The last entered number is flashed.
7. Having more than 9 open parentheses or more than 4 pending operations. The 10th parenthesis or 5th operation is not accepted so calculation can continue. The last displayed number is flashed.
8. Dividing a number by zero. "9.9999999 99" is flashed. (Except 0 $\boxed{\div}$ 0 $\boxed{=}$ 1 flashing.)
9. Pressing an operation key before completing any memory operation, fix decimal operation or direct transfer operation. The value in the display is flashed. (i.e. $\boxed{\text{STO}}$ $\boxed{+}$)
10. Attempting to store, recall, or use other memory operations with locations other than 0 through 7. (i.e. $\boxed{\text{STO}}$ 8).
11. Attempting to find values of functions with entered values outside these limits:

Function	*Limit*
$\sin^{-1}x$, $\cos^{-1}x$	$-1 \le x \le 1$
e^x	$-227.95592 \le x \le 230.25850$
10^x	$-99 < x < 100$
ln x	$1 \times 10^{-99} \le \lvert x \rvert < 1 \times 10^{100}$
log x	$1 \times 10^{-99} \le x < 1 \times 10^{100}$
$\boxed{\text{INV}}$ $\boxed{\text{P}\rightarrow\text{R}}$	$1 \times 10^{+50} < R$

12. Direct transfer instructions (GTO or SBR) that attempt to branch to unassigned label positions. Current display value is flashed. A GTO 2nd **nn** key sequence where **nn** is greater than 49 also produces a flashing condition.
13. Attempting to execute a program past location 49. Current display value is flashed.
14. Attempting to calculate $\bar{x}$ or σ^2 without entering any data or only entering a single data point.

When Errors Are Encountered When Running a Program:
When any of the foregoing errors occur in a program, the program will stop and show you a flashing display. This means you've asked the calculator to do something in a program that it cannot do — even directly from the keyboard.

APPENDIX D: DISPLAYED RESULTS VS ACCURACY

The basic mathematical tolerance of the calculator is controlled by the number of digits it uses for calculations. The calculator appears to use 8 digits as shown by the display, but actually uses 11 digits to perform all calculations. Combined with the built-in 5/4 rounding capability, these extra digits guard the 8-digit display to improve accuracy. Consider the following example in the absence of these guard digits.

⅓ × 3 = .99999999 (inaccurate)

The example shows that 1 ÷ 3 = .33333333 when multiplied by 3 produces an inaccurate answer. However, an eleven-digit string of nines will round to 1 when rounded to 8 places.

The higher order mathematical functions use interative calculations. The cumulative rounding error is usually maintained below the 8-digit display so that no effect can be seen. The three "guard digits" act to protect the display from small cumulative errors.

Normally, there is no need to even consider these guard digits. In certain calculations, however, the guard digits may appear as an answer when not expected. Results of two computations may appear to be equal when shown in the display, but have an inequality in their guard digits. If these two results are subtracted, the calculator may display a nonzero result.

This fact is especially important when writing your own programs. If you're testing a calculated result to be equal to another value, such as with the 2nd x=t instruction, you may need to take precautions to prevent improper evaluation due to the guard digit differences. The key sequence EE INV EE will truncate the guard digits of a result leaving only the rounded display value for further use.

For the standard display, results are accurate for all calculations that do not violate the restrictions listed in Appendix C, except as defined below.

TRIGONOMETRIC FUNCTIONS

All displayed digits are accurate for a ±36,000 degree range (±200 π radians and ±40,000 grads). In general, the accuracy decreases one digit for each decade outside this specified accuracy range. An exception is the tangent of an odd multiple of ±90°, ±π/2 radians or ±100 grads that results in an overflow condition because the function is undefined at these points.

ROOTS AND POWERS

There can be some accuracy loss for roots and powers in calculations when the base (y) gets very close to 1 and the power (x) gets very large.

APPENDIX E: KEY CODES

CALCULATOR KEY CODES — USING ACTUAL KEYBOARD ARRANGEMENT

Rows ↓					
1	2nd No Code	INV Minus Sign	log 18 lnx 13	C.t 19 CE 14	CLR 15
2	D.MS 26 LRN (No Code)	P→R 27 x⇌t 22	sin 28 x^2 23	cos 29 $\sqrt{x}$ 24	tan 20 1/x 25
3	Pause 36 SST (No Code)	Ins (No Code) STO 32	Exc 38 RCL 33	Prd 39 SUM 34	π 30 y^x 35
4	Nop 46 BST (No Code)	Del (No Code) EE 42	FIX 48 (43	Int 49) 44	|x| 40 ÷ 45
5	Dsz 56 GTO 51	7 07	8 08	9 09	Deg 50 × 55
6	x=t 66 SBR 61	4 04	5 05	6 06	Rad 60 − 65
7	x≥t 76 RST 71	1 01	2 02	3 03	Grad 70 + 75
8	Lbl 86 R/S 81	0 00	Σ+ 88 • 83	$\bar{x}$ 89 +/− 84	σ2 80 = 85
Columns	1	2	3	4	5
(for → second functions)	6	7	8	9	0

CALCULATOR KEY CODES IN NUMERICAL ORDER

00	0
01	1
02	2
03	3
04	4
05	5
06	6
07	7
08	8
09	9
13	lnx
−13	INV lnx
14	CE
15	CLR
18	2nd log
−18	INV 2nd log
19	2nd C.t
−19	INV 2nd C.t
20	2nd tan
−20	INV 2nd tan
22	x⇌t
23	x^2
24	$\sqrt{x}$
25	1/x
26	2nd D.MS
−26	INV 2nd D.MS
27	2nd P→R
27	INV 2nd P→R
28	2nd sin
−28	INV 2nd sin
29	2nd cos
29	INV 2nd cos
30	2nd π
32	STO
33	RCL
34	SUM
34	INV SUM
35	y^x
−35	INV y^x
36	2nd Pause
38	2nd Exc
39	2nd Prd
39	INV 2nd Prd
40	2nd |x|
42	EE
−42	INV EE
43	(
44	)
45	÷
46	2nd Nop
48	2nd FIX
−48	INV 2nd FIX
49	2nd Int
−49	INV 2nd Int
50	2nd Deg
51	GTO
55	×
56	2nd Dsz
−56	INV 2nd Dsz
60	2nd Rad
61	SBR
61	INV SBR
65	−
66	2nd x=t
−66	INV 2nd x=t
70	2nd Grad
71	RST
75	+
76	2nd x≥t
−76	INV 2nd x≥t
80	2nd σ2
−80	INV 2nd σ2
81	R/S
83	•
84	+/−
85	=
86	2nd Lbl
88	2nd Σ+
−88	INV 2nd Σ+
89	2nd $\bar{x}$
−89	INV 2nd $\bar{x}$

TYPICAL ANSWERS TO "NEXT STOP" PROBLEMS

(*Note:* In general, there is no one "right" program answer to any problem situation. We will show one approach here.)

On Sale!
page 4-1

```
OFF/ON
LRN
×  (  1  −
RCL  3  )  =
STO  5
1  SUM  4
RCL  5
R/S
RST
LRN  RST
```

Count Off
page 4-3

```
OFF/ON  LRN
STO  1  R/S
STO  2
2nd  Lbl  1
RCL  1  y^x
RCL  3  =
2nd  Pause
2nd  Pause
RCL  2  SUM  3
GTO  1
LRN  RST
```

Factorial!
page 4-6

```
OFF/ON  LRN
2nd  Lbl  2
1  SUM  2  RCL  2
2nd  Pause  2nd  Pause
STO  0
2nd  Lbl  1
RCL  0  ×
2nd  Dsz  GTO  1
1  =  2nd  Pause
2nd  Pause  2nd  Pause
GTO  2
LRN  RST
```

(GB)

KEEP THE CARD ON THE REVERSE SIDE
(PROOF-OF-PURCHASE DATE)

- IMPORTANT : The only applicable warranty conditions are those described in your Owner's Manual.

Please fill in the attached card and send it back to your closest Texas Instruments office. (See address on the back side cover of your owner's manual).

(F)

CONSERVEZ AVEC SOIN LA CARTE IMPRIMEE AU DOS
(JUSTIFICATION DE LA DATE D'ACHAT)

- IMPORTANT : Les seules conditions de garantie applicables sont celles décrites dans votre manuel d'utilisation.

Veuillez remplir la carte ci-contre et la renvoyer au bureau le plus proche de Texas Instruments (adresse sur le dos de la couverture de votre manuel d'utilisation)

(D)

BEWAHREN SIE DIE RÜCKSEITIGE KARTE SORGFÄLTIG AUF.
(NACHWEIS ÜBER DAS KAUFDATUM)

- WICHTIG : Es gelten nur die in Ihrer Gebrauchsanweisung abgedruckten Gewährleistungsbedingungen.

Bitte die anhängende Karte sorgfältig ausfüllen und sofort an die für Sie zuständige Texas Instruments Niederlassung absenden (Adresse siehe Rückseite Bedienungsanleitung).

(I)

CONSERVATE CON CURA LA SCHEDA STAMPATA SUL RETRO.
(PROVA DELLA DATA DI ACQUISTO).

- IMPORTANTE : Le sole condizioni di garanzia valide sono quelle descritte sul vostro libretto d'istruzioni.

Preghiamo riempire la scheda allegata e spedirla al più vicino ufficio della Texas Instruments (indirizzo sul retro della copertina del manuale d'istruzione).

(NL)

BEWAAR DE KAART AAN OMMEZIJDE ZORGVULDIG
(BEWIJS VAN AANKOOPDATUM).

- BELANGRIJK : Van toepassing zijn uitsluitend de in uw gebruiksaanwijzing vermelde garantievoorwaarden.

Gelieve de aangehechte kaart, volledig ingevuld, op te zenden aan het voor u dichtstbijzijnde kantoor van Texas Instruments (zie adressen op achterzijde van uw gebruiksaanwijzing).

(D) (I)

TEXAS INSTRUMENTS

GEWÄHRLEISTUNGSKARTE ~ SCHEDA DI GARANZIA

Modell / Modello TI-57 Serien Nr. / Numero di serie ____ Datum : / Data : ____

1 ☐ Herr, Sig. re 2 ☐ Frau, Fräulein, Sig. ra, Sig. na 3 ☐ Firma, Ditta

Familienname / Cognome ____ Vorname / Nome ____

Firma : / Ditta : ____

Straße : / Indirizzo : ____

Ort : / Città : ____ Postleitzahl : / Codice postale : ____ Land : / Paese : ____

War der Rechner ein Geschenk ? / La calcolatrice è stata regalata ? 1 ☐ Ja / Si 2 ☐ Nein / No

Gebrauch / Uso 1 ☐ Privat / Personale 2 ☐ Geschäftlich / Professionale 3 ☐ Beides / Ambedue

WO GEKAUFT ?
LUOGO DI ACQUISTO ?

- A ☐ Kaufhaus / Grande magazzino
- B ☐ Grosseinkauf / Ipermercato
- C ☐ Bürofachhändler / Negozio di articoli per ufficio
- D ☐ Rechner-Spezialgeschäft / Negozio specializzato in Calcolatrici
- E ☐ Versandhandel / Per posta
- F ☐ Elektrogeschäft / Negozio elettrodomestici
- G ☐ Radiogeschäft / Negozio Radio TV
- H ☐ Fotogeschäft / Negozio cinefoto
- I ☐ Schreibwarengeschäft / Cartoleria
- J ☐ Andere (spezifizieren) / Altro (Indicare)

Besitzen Sie schon einen Rechner ? / Avete già un'altra calcolatrice ? 1☐ ja / si 2☐ nein / no

IHR BERUF
LA VOSTRA PROFESSIONE

- A ☐ Ingenieur / Ingegnere
- B ☐ Wissenschaftler / Scienziato
- C ☐ Geschäftsmann / Uomo d'affari
- D ☐ Buchhalter / Ragioniere
- E ☐ Schüler / Studente (liceo)
- F ☐ Student / Studente (università)
- G ☐ Lehrer, Dozent / Insegnante, Professore
- H ☐ Ladeninhaber / Commerciante
- I ☐ Arzt - Rechtsanwalt / Medico - Legale
- J ☐ Techniker / Tecnico
- K ☐ Architekt - Landvermesser / Architetto - Geometra
- L ☐ Finanzfachmann / Finanziario
- M ☐ Andere (spezifizieren) / Altro (Indicare) ____

IHR ALTER
LA SUA ETA

- 1 ☐ — 14
- 2 ☐ 15 - 18
- 3 ☐ 19 - 28
- 4 ☐ 29 - 48
- 5 ☐ 49 +

TEXAS INSTRUMENTS

WARRANTY CARD • CARTE DE GARANTIE

GEWÄHRLEISTUNGSKARTE • GARANTIEKAART

PROOF-OF-PURCHASE DATE
JUSTIFICATIF DE LA DATE D'ACHAT
NACHWEIS ÜBER DAS KAUFDATUM
PROVA DELLA DATA DI ACQUISTO
AANKOOPDATUMBEWIJS

Date, Date, Datum, Data, Datum. ____________

Retailer Stamp
Cachet du revendeur
Händler Stempel
Timbro del rivenditore
Stempel wederverkoper

Model / Modèle / Modell / Modello / Model: TI-57

Serial N° / N° de série / Serien Nr. / Numero di serie / Serienummer: ____________

1 ☐ MR. / Monsieur / Herr / Sig. re / Dhr.

2 ☐ Miss, Mrs. / Mme, Melle. / Frau, Fräulein / Sig. ra, Sig. na / Mevr. Mej.

3 ☐ Company / Société / Firma / Ditta / Firma

Last Name / Nom / Familienname / Cognome / Achternaam: ____________

First Name / Prénom / Vorname / Nome / Voornaam: ____________

Company / Société / Firma / Ditta / Firma: ____________

Address / Adresse / Adresse / Indirizzo / Adres: ____________

Town / Ville / Ort / Città / Stad: ____________

P. O. Code / Code Postal / Postleitzahl / Codice Postale / Postcode: ____________

Country / Pays / Land / Paese / Land: ____________

Please describe failure
Description de la panne
Bitte, beschreiben Sie den gefundenen Fehler
Descrizione del guasto
Korte beschrijving van de storing

KEEP CONSERVEZ BEWAHREN SIE CONSERVATE BEWAREN

(GB) WARRANTY CARD (F) CARTE DE GARANTIE (NL) GARANTIE KAART

TEXAS INSTRUMENTS

Model / Modèle / Model: TI-57

Serial N° / N° de série / Serie nr.: ____________

Date : / Date : / Datum : ____________

1 ☐ MR. Monsieur, Dhr. 2 ☐ Miss, Mrs, Mme, Melle, Mevr., Mej. 3 ☐ Company, Société, Firma

Last Name / Nom / Achternaam: ____________

First Name / Prenom / Voornaam: ____________

Address : / Adresse : / Adres : ____________

Town : / Ville : / Stad : ____________

P.O. Code : / Code Postal : / Postcode : ____________

Country : / Pays : / Land : ____________

Was calculator a gift ? / S'agit-il d'un cadeau ? / Was rekenmachine een kadoo ?
Yes / Oui / Ja 1 ☐ 2 ☐ No / Non / Nee

Use / Utilisation / Gebruik
1 ☐ Personal / Personnelle / Persoonlijk
2 ☐ Professional / Professionnelle / Beroepshalve
3 ☐ Both / Les deux / Beide

WHERE PURCHASED ?
LIEU D'ACHAT ?
WAAR GEKOCHT ?

- A ☐ Dept. Store / Grand magasin / Warenhuis
- B ☐ Hypermarket / Grande Surface / Hypermarkt
- C ☒ Office equipment dealer / Magasin d'équipement de bureau / Kantoorboekhandel
- D ☐ Calculator shop / Magasin de calculatrices / Speciaalzaak v. rekenmachines
- E ☐ Mail order / Par correspondance / Postorder
- F ☐ Home appliance shop / Magasin d'électro-ménager / Winkel huishoudelijke app.
- G ☐ Radio TV shop / Magasin Radio Télévision / Radio-/Televisiezaak
- H ☐ Camera shop / Magasin photo / Fotohandel
- I ☐ Stationery shop / Librarie - Papeterie / Boekhandel
- J ☐ Other (Specify) / Autre (Préciser) / Elders (Toelichten)

Do you already own a calculator ? / Possédez-vous une autre calculatrice ? / Bezit u al een rekenmachine ?
Yes / Oui / Ja 1 ☐ 2 ☐ No / Non / Nee

YOUR OCCUPATION
VOTRE PROFESSION
UW BEROEP

- A ☐ Engineer / Ingénieur / Ingenieur
- B ☐ Scientist / Chercheur / Wetenschapper
- C ☐ Business man / Homme d'affaires / Zakenman
- D ☐ Accountant / Comptable / Boekhouder
- E ☐ College student / Lycéen / Scholier
- F ☐ University student / Etudiant / Student
- G ☐ Teacher, Professor / Professeur / Onderwijzer
- H ☐ Shop keeper / Commerçant / Winkelier
- I ☐ Physician/Lawyer / Médecin/Juriste / Dokter/Jurist
- J ☐ Technician / Technicien / Technicus
- K ☐ Architect/Surveyor / Architecte/Géomètre / Architect/Landmeter
- L ☐ Financier / Financier / Financier
- M ☐ Other (Specify) / Autre (Préciser) / Andere (toelichten)

YOUR AGE
VOTRE AGE
UW LEEFTIJD

- 1 ☐ – 14
- 2 ☐ 15 - 18
- 3 ☐ 19 - 28
- 4 ☐ 29 - 48
- 5 ☐ 49 +

ONE-YEAR-LIMITED WARRANTY

The TI-57 electronic calculator (including charger) from Texas Instruments is warranted to the original purchaser for a period of one (1) year from the original purchase date — under normal use and service — against defective materials or workmanship. **ANY IMPLIED WARRANTIES ARE ALSO LIMITED IN DURATION TO THE ONE YEAR PERIOD FROM THE ORIGINAL PURCHASE DATE.**

This warranty is void if : the calculator has been damaged by accident or unreasonable use, neglect, improper service or other causes not arising out of defects in material or workmanship.

TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR LOSS OF USE OF THE CALCULATOR OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS EXPENSES OR DAMAGES INCURRED BY THE PURCHASER.

During the above one year period, the calculator or its defective parts will be repaired, adjusted and/or replaced with reconditioned model of equivalent quality, ("REFURBISHED"), at Manufacturer's option, without charge to the purchaser when the calculator is returned, prepaid and insured, with proof-of-purchase date, to Texas Instruments. **UNITS RETURNED WITHOUT PROOF-OF-PURCHASE DATE WILL BE REPAIRED AT THE SERVICE RATES IN EFFECT AT THE TIME OF RETURN.**

In the event of replacement with a reconditioned model, the replacement unit will continue the warranty of the original calculator product or 90 days whichever is longer.

IMPORTANT : Before returning your calculator for repair, carefully review service and mailing instructions in this manual.

SERVICE CENTERS

BELGIE - BELGIQUE
Avenue Edouard Lacomblé, 21
1040 - Bruxelles Brussel
Tel. (2) 733 96 23

CANADA
41 Shelley Road
Richmond Hill, Ontario
Tel. (416) 889 73 73

DANMARK
Marielundvej 46 E
2730 Herlev
Tel. (02) 91 74 00

DEUTSCHLAND
Kepserstraße 33
8050 - Freising
Tel. (08161) 7411

ENGLAND
Manton Lane
Bedford, MK41 7PU
Tel. (0234) 67466

ESPAÑA
Apartado 98
Torrejon de Ardoz - Madrid
Tel. 675 53 00
675 53 50

FRANCE
B.P. 28
06021 - Nice Cedex

ITALIA
Casella Postale 1
02015 - Cittaducale
Tel. (0746) 690 34/35/36

NEDERLAND
Postbus 43
Kolthofsingel 8
Almelo
Tel. (05490) 63967

NORGE
Ryensvingen 15
Oslo 6
Tel. (02) 68 94 87

ÖSTERREICH
Marxergasse 10
1030 - Wien
Tel. (0222) 72 41 86

PORTUGAL
2650 Rua Eng° ,
Frederico Ulrich
Moreira Da Maia
Douro
Tel. (02) 948 1003

SCHWEIZ - SUISSE
Aargauerstraße 250
8048 - Zürich
Tel. (01) 64 34 55/56

SUOMI FINLAND
P.L. 917
Freesenkatu 6
00101 - Helsinki 10
Tel. (80) 40 83 00

SVERIGE
Norra Hamnvägen 3
Fack
100 54 Stockholm 39
Tel. (08) 23 54 80

TEXAS INSTRUMENTS